Data Integration with XML and Semantic Web Technologies

A thesis presented

by

Rubén Tous

Submitted in partial fulfilment of the requeriments for
the degree of Doctor of Philosophy
Doctorate in Computer Science and Digital Communication
Department of Technologies

Advisor: Catedràtic d'Universitat Jaime Delgado Mercé

Universitat Pompeu Fabra

Barcelona, June 2006

Dipòsit legal: B.48973-2006
ISBN: 978-84-690-3512-2

Thesis advisor
Catedràtic d'Universitat Jaime Delgado Mercé

Author
Rubén Tous

Data Integration with XML and Semantic Web Technologies

Abstract

Accessing data from multiple heterogeneous databases entails dealing with different data models, different schemas and different query languages and interfaces. This thesis can be divided in two main parts, both related to different aspects of the design of modern data integration systems. The first part is related to the problem of dealing with heterogeneous data models and schemas. It is also divided in two different sub-parts. The first one is focused in the semantic integration between XML and RDF (the Resource Description Framework). We suggest a strategy based on the mapping of the XML model to RDF triples. Translating XML documents to RDF permits taking profit from the powerful tools of Description Logics allowing XML documents interoperate at the semantic level. This mapping has generated some interesting results, like a schema-aware and ontology-aware XPath processor that can be used for schema semantic integration or even for implicit query transcoding among different data models. The approach have been tested in the Digital Rights Management (DRM) domain, where some organizations are involved in standardization or adoption of rights expression languages (REL).

In the second sub-part we suggest a vector space model for semantic similarity measurement and OWL ontologies alignment. A rigorous, efficient and scalable similarity measure is a pre-requisite of any ontology matching system. The presented model is based on a matrix representation of nodes from an RDF labelled directed graph. A concept is described with respect to how it relates to other concepts using n-dimensional vectors, being n the number of selected common predicates. We have successfully tested the model with the public testcases of the Ontology Alignment Evaluation Initiative 2005.

The second part of the work is related to the problem of dealing with heterogeneous query interfaces. We suggest a strategy that allows redistributing an expressive user query (expressed in a XML-based data query language) over a set of autonomous and heterogeneous databases accessed through web forms. The idea, that has recently been renamed by Thomas Kabisch [81] as "Query Tunneling", consists on the reprocessing of the initial user query over the results returned by the different sources, that must be a superset of the results that satisfy the initial query. We describe in this document the strategy and its limitations, and an implementation in the form of two Java APIs, the Java Simple API for Metasearch (JSAM), that has been used in the development of a spanish news metasearch engine, and the Java Simple API for Web Information Integration (SAWII), that offers high level tools to the development of articulated wrappers for complex web form-chains and result pages.

Contents

List of Figures

List of Tables

Already Published Work

Large portions of Chapter 9 have appeared in the following papers:

Tous R., García R., Rodríguez E., Delgado J. “Architecture of a Semantic XPath Processor. Application to Digital Rights Management”, 6th International Conference on Electronic Commerce and Web Technologies EC-Web 2005. August 2005 Copenhagen, Denmark. Lecture Notes in Computer Science, Vol. 3590 (2005), pp. 1-10. ISSN: 0302-9743

Tous R., Delgado J. “A Semantic XPath processor”. InterDB 2005 International Workshop on Database Interoperability. ELSEVIER’s Electronic Notes in Theoretical Computer Science 2005

Tous R., Delgado J. “RDF Databases for Querying XML. A Model-mapping Approach”. DISWeb 2005 International Workshop Data Integration and the Semantic Web. Procedings of the CAiSE’05 Workshops. Faculdade de Engenharia da Universidade do Porto. ISBN 972-752-077-4 Pages: 363 - 377

Tous R., Delgado J. “Using OWL for Querying an XML/RDF Syntax”. WWW’05: Special interest tracks and posters of the 14th international conference on World Wide Web. Chiba, Japan. Pages: 1018 - 1019. ACM Press 2005. ISBN:1-59593-051-5. http://doi.acm.org/10.1145/1062745.1062847

Large portions of Chapter 10 have appeared in the following paper:

Tous R., Delgado J. “A Vector Space Model for Semantic Similarity Calculation and OWL Ontology Alignment”, 17th International Conference on Database and Expert Systems Applications (DEXA 2006), 4-8 September 2006. To be published in Lecture Notes in Computer Science.

Large portions of Chapters 11 and 12 have appeared in the following papers:

Gil R., Tous R., García R., Rodríguez E., Delgado J. “Managing Intellectual Property Rights in the WWW: Patterns and Semantics”. 1st International Conference on Automated Production of Cross Media Content for Multi-channel Distribution (AXMEDIS 2005), November 2005

Tous R., Delgado, J. “Interoperability Adaptors for Distributed Information Search on the Web”. Proceedings of the 7th ICCC/IFIP International Conference on Electronic Publishing 2003. http://elpub.scix.net/cgi-bin/works/Show?0341

Tous R., Delgado, J. “Advanced Meta-Search of News in the Web”, Proceedings of the 6th International ICCC/IFIP Conference on Electronic Publishing. Publisher: VWF Berlin, 2002. ISBN 3-89700-357-0. 395 pages. http://elpub.scix.net/cgi-bin/works/Show?0234

Electronic preprints are available on the Internet at the following URL:

http://dmag.upf.edu/∼rtous

Acknowledgments

This work has been supported by Agent Web, TIC 2002-01336 Spanish Administration project, and by VISNET, European Commission IST FP6 Network of Excellence. I would like also to express my gratitude to Jaime Delgado, my Advisor, without whom this thesis have never begin and what is worst, never end. I am also grateful to Miquel Oliver and Joan Vinyes for their support in difficult times. I thank Juan C. Sánchez for sharing with me so good and bad moments during the Doctorate Programme. I thank Ana Escudero for being such a patient deskmate and for her grammar wisdom. I thank Boris Bellalta for his sense of humour and sarcastic point of view. I thank Anna Sferopoulou for her unconditional support and inspiration. I thank Mohammad Mobaseri for his quiet and meditated point of view. I thank Jaume Barceló for his impulsive and extreme point of view (that's a cross acknowledgment ;-)) I thank Roberto García for introducing me to the Semantic Web and for our breakfast brainstorms, undoubtfully one of my best sources of inspiration. I thank Eva Rodríguez for her practical opinions and her contributions in the DRM part. I thank Rosa Gil for her sincerity and sensibility. I thank Manel Guerrero for his comments and help with Linux. I'm in real debt with Ramon Martí for his help and partnership in my beginnings at UPF. I thank Jorge Infante for our neverending discussions where I never win but always learn. I thank Núria Garcia for her experienced counsel, and her humanity. I thank Susanna 'Belén' Fernández, Iolanda Sabater and Carina Ortega for making my visits to the ESUP secretariat a pleasure. I thank Roser Clavero for being the nicest and most competent professional I ever met. I am also grateful to all my other friends from the Department of Technology (present and former). I am forever indebted to my parents, my brother and Elena for their understanding, endless patience and encouragement.

Dedicated to Lourdes, Benjamí, Dan and Elena

Chapter 1

Introduction

This introductory chapter serves to three purposes. In one hand to define the scope of the work that is addressed by this document. In the other hand to facilitate the reading of the document by giving and overview of its structure. Finally, but the most important, to provide the necessary elements to enclose the work in the framework of a PhD thesis.

1.1 About this thesis

This document describes research work being done under the framework of the Distributed Multimedia Applications Group (DMAG), and also in the context of the Doctorate in Computer Science and Digital Communication of the Department of Technology of the Universitat Pompeu Fabra. The work focuses on the different aspects related to the design of modern data integration systems in the context of the World Wide Web, combining an exploratory stage with others focused on the design of new strategies and models and the development of specific tools. The main scope of the work is the data integration problem, within the databases research discipline, but that also falls between other well-known disciplines. On one hand Information Retrieval, since in most cases the purpose of storing and accessing data and metadata is not the metadata themselves but the support to information search and retrieval processes. On the other hand the Web discipline, and specially the Semantic Web Initiative, a promising and relatively new research line where metadata has a central role. Among all these sources I've tried to choose the elements that define the context of this work, and the weaknesses and opportunities that justify the work itself.

1.2 Aims and Hypothesis

The general goal of this research project is to study solutions to the new problems arisen with respect to the querying of distributed and heterogeneous sources of data and metadata. This goal can be seen as a reformulation of some topics faced by the data integration community, but now arisen again as databases become open and accessible through the Web, storing data and metadata in old and new formats that can be the basis of better ways of searching and retrieving digital contents. Within this ambitious and broad aim, the work has focused in two related but independent aspects, semantic integration and query interoperability. Within the semantic integration problem, we face the XML-RDF integration from a novel approach, and also the ontology alignment problem. Within the query interoperability problem, we face the querying of restricted and heterogeneous web-based database interfaces with XML technologies.

A more tangible sub-goal of the work is the development of tools (in the form of APIs) to test the models and strategies suggested. For the first part, related to semantic integration, the implementation of a Semantic XPath processor and its usage in the Digital Rights Management (DRM) domain will demonstrate the interest of our novel way to map XML and RDF. Related to ontology alignment, the implementation of our novel structure similarity measure and its results against the Ontology Evaluation Initiative 2005 testsuite will demonstrate the relevance of our approach. For the second part, related to query interoperability, two Java APIs - the Java Simple API for Metasearch (JSAM) and the Java Simple API for Web Information Integration (SAWII) - integrated inside an advanced spanish news metasearch engine will serve to prove the advantage of the Query Tunneling technique.

All the work is sustained over the presumption that the traditional data integration strategies still don't take profit from the potential of the new uses of metadata on the Web and some of its new directly or indirectly related technologies, like XML [151], XML Query [154], RDF [124] or OWL [112]. The success of standard syntaxes for metadata (XML and RDF), the dissemination of metadata related to multimedia contents (e.g. MPEG-7 [101]) or to the more broad framework of the Semantic Web, open new opportunities and challenges for distributed data retrieval and data integration.

For a more detailed description of the goals of the thesis please refer to the *Problem Statement* chapter.

1.3 Methodology

In order to avoid replication of work, and also to achieve the necessary background, the exploratory and analytical stage of the thesis has implied the identification and reading of relevant materials from a lot of different sources. Because the work falls between different research disciplines, that sometimes have different approaches to the same problems, I have tried to identify the most authoritative sources of each field.

1.3.1 Related to the first contribution part 'Heterogeneous Data Models and Schemas: Semantic Integration'

I acquired background of issues related to semantic integration in the works of A. Y. Halevy [54] et al. (the Piazza Infrastructure), I. Cruz et al. [65], B. Amann et al. [5], M.C.A.Klein, L.V.Lakshmanan and F.Sadri [87], P.F.Patel-Schneider and J.Simeon [117]. However, the spark that inspired the contribution of the first sub-part (XML-RDF integration) came from the reading of a relatively old work of Yoshikawa et al. [97] that differentiated between approaches that map the structure of some XML schema to a set of relational tables and works that map the XML model to a general relational schema respectively. I've also found a previous work from Lehti and Fankhauser [91] that also pursues the target to achieve a *semantic* behaviour for XPath/XQuery. For the formalisation of the XML/RDF Syntax with description logics I've borrowed abundant material from Ian Horrocks and Ulrike Sattler, like [62], [60], [59], [63] and [61]. During the development of the inference-based XPath processor I've used the Jaxen Universal Java XPath Engine [75] for parsing XPath 2.0 expressions and the Jena API [76] and its OWL inference engine [77] for processing the queries. Among the differnt RDF query languages (rdfDB [125], SquishQL [139], RDQL [127], or the recent SPARQL [138]) I've chosen RDQL for its maturity and because it's supported by the Jena API. This stage has entailed also frequent visits to some unfriendly W3C specifications like the XML Information Set [68], the XML Path Language (XPath) 2.0 [69], XQuery 1.0 [154], XQuery 1.0 and XPath 2.0 Data Model [152], XQuery 1.0 and XPath 2.0 Formal Semantics [153], RDF/XML Syntax Specification [147], OWL Web Ontology Language Overview [112] and XSL Transformations (XSLT) Version 2.0 [155].

For the second sub-part (a vector space model for sematic similarity calculation), I've found a good survey on schema alignment from Erhard Rahm and Philip A. Bernstein [122], and a more recent survey on ontology alignment from Natalya F. Noy [106]. I've also read abundant material about semantic similarity, like the old works of Philip Resnik [129], Dekang Lin [94] or J.J. Jiang and D.W. Conrath [79]; and also more recent ones like those of Francisco Azuaje and Olivier Bodenreider [8] or M. Bisson [19] among others. Here the key work where I've found inspiration has been the paper of Wei Hu et al. [64] from who I've also received personal support and patient clarifications. The graph matching algorithm that I have used comes from the work of Vincent D. Blondel et al. [20].

1.3.2 Related to the second contribution part 'Heterogeneous Query Interfaces: Query Tunneling'

For general concepts on the Information Retrieval area I've borrowed a lot of material from the work of Ricardo Baeza [9] and some of the references he uses. For more specific aspects on Web search, I've studied the works of Kobayashi and Takeda [85], Lawrence and Giles [88], Brin and Page [23][113] and others. It has been difficult to find rigurous research works about metasearch, because is a popular topic among independent and sometimes volunteer-based communities, but I've found a good basis on the works of Dreilinger [33] and Selberg and Etzioni [135], among others.

For the study about the Semantic Web I've read materials by the people who are behind this initiative, like Tim Berners-Lee [11][15][12][14][13], James Hendler [17], Ora Lassila [17] or Dan Brickley [22], and I've had to assimilate the specification of RDF and other related technologies. I've also invested some time trying to understand and appreciate the added value of the RDF model against others (e.g. relational or XML), and I have found some good materials about this topics, like e.g. [25] or [13]. Searching for materials related to the Semantic Web activity, I've found some promising works, like the RDFWeb initiative and its FOAF ontology [39], the work of Guha and MacCool [120][49] or the Query by Example by Reynolds [130]. To make the analysis about digital libraries and general metadata issues I've spent some time studying the specifications of the Dublin Core Element Set [28], the Z39.50 protocol [156], SRW [140] and the OAI-PMH protocol [109][110]. I've also found some works discussing issues about digital libraries and the Open Archives Initiative, like e.g. those by Lagoze and Sompel [86][137] or Baker [10]. These are just some of the materials reviewed, but serve to illustrate in some way the selection criteria.

For the development of the specialised metasearch application I've followed a traditional software engineering approach. The requirements stage has consisted in innumerable meetings with my advisor, Jaime Delgado, to try to define clearly the different aspects of the functionality. The analysis stage has involved the hard task of translating the natural language requirements to some formal model, I've used UML to do this but also some home-made diagrams. The design stage has consisted in weaving the objects architecture of the application, with an special emphasis in the multi-threading sub-parts. It has also involved the specification of test XML-based query language and the study of the critical parts of the system, as the mapping of the queries or the metadata extraction. I've studied to do this some existing XML-based Query Languages, like XCQL [150] or XML Query [154]. The implementation methodology has involved the study and selection of the tools (Java, Tomcat, XML, XML Query, Tidy) and the analysis of the performance, where I've invested an important time, specially in defining a good policy to discard targets that are suffering of not ordinary delays. I'm going to talk more about the implementation methodology later.

1.4 Document outline

This document has four main parts, 'Background Information', 'State of the Art and Problem Statement', 'Heterogeneous Data Models and Schemas: Semantic Integration' and 'Heterogeneous Query Interfaces: Query Tunneling', and also other smaller sections like this introduction or the final comments.

1.4.1 Background Information

The 'Background Information' part tries to compile in a coherent way some results of the exploratory stage that can help readers not familiar with databases, information retrieval or the semantic web. This implies to define and describe the key concepts of these areas and related to the work, and also the difficult task to weave all the relationships between them. Readers familiar with these technologies will probably skip this part.

1.4.2 State of the Art and Problem Statement

The 'State of the Art and Problem Statement' part is a survey of the progresses being done in the data integration field in general and the semantic integration research trend in particular in the last years. It begins enumerating some classical works and concepts of the area and ends analysing its current weaknesses and opportunities.

1.4.3 Heterogeneous Data Models and Schemas: Semantic Integration

After the State of the Art come two related but independent parts containing the main contributions of this work. Because of their particularities each one of them provides a complete internal structure, including an introduction, a related work section and conclusions.

The first main contribution, 'Heterogeneous Data Models and Schemas: Semantic Integration' part includes two works related to the use of Semantic Web technologies to help solving the problem to work in domains where multiple schemas or ontologies exist. The first sub-part, 'XML Semantic Integration: A Model Mapping Approach' , describes a model to face the problem of dealing with heterogeneous data models and schemas. It is focused in the semantic integration between XML and RDF (the Resource Description Framework). I suggest a strategy based on the mapping of the XML model to RDF triples. Translating XML documents to RDF permits taking profit from the powerful tools of Description Logics allowing XML documents interoperate at the semantic level. This mapping has generated some interesting results, like a schema-aware and ontology-aware XPath processor that can be used for schema semantic integration or even for implicit query transcoding among different data models.

The second sub-part, 'A Vector Space Model for Semantic Similarity Calculation and OWL Ontology Alignment', describes a novel semantic similarity measure based on a matrix representation of nodes from an RDF labelled directed graph. A concept is described with respect to how it relates to other concepts using n-dimensional vectors, being n the number of selected common predicates. It shows how adapting the graph matching algorithm in [20] to apply this idea to the alignment of two ontologies. It also includes the results of the testcases of the Ontology Alignment Evaluation Initiative 2005.

1.4.4 Heterogeneous Query Interfaces: Query Tunneling

The second main contribution, 'Heterogeneous Query Interfaces: Query Tunneling' describes a strategy to face the problem of dealing with heterogeneous query interfaces. I suggest a strategy that allows redistributing an expressive user query over a set of databases with heterogeneous Web-based query interfaces. The idea, that has recently been renamed by Thomas Kabisch [81] as "Query Tunneling", consists on the reprocessing of the initial user query over the results returned by the different sources, that must be a superset of the results that satisfy the initial query. I describe in this part the strategy and its limitations, and an implementation in the form of two Java APIs, the Java Simple API for Metasearch (JSAM), that has been used in the development of a spanish news metasearch engine, and the Java Simple API for Web Information Integration (SAWII), that offers high level tools to the development of articulated wrappers for complex web form-chains and result pages.

1.4.5 Relationship between thesis parts and chapters

The first part, 'Heterogeneous Data Models and Schemas: Semantic Integration', refers to the problem of schema/ontology interoperability. This problem can be divided in two sub-problems, 1) How can we use a global (mediator) data schema/ontology to interoperate a set of heterogeneous schemas/ontologies? and 2) How can we automatically generate this global schema/ontology?

Chapter 9 faces the first of these problems for the particular case of XML, suggesting a strategy to design a schema-aware and ontology-aware XPath processor, which process queries taking in consideration the relationships defined in one or more XML schemas or OWL/RDFS ontologies. This allows to write the queries in terms of one of the schemas/ontologies (acting as a global schema), whose relationships with the specific schemas have been previously specified. How these relationships (mappings) are obtained is not the focus of this chapter, but of the next one.

Chapter 10 is independent from the previous one, but is related to the second problem (how the mappings between ontologies/schemas can be automatically generated). This is the reason why the two chapters have been placed in the same part.

The second part, 'Heterogeneous Query Interfaces: Query Tunneling', describes a practical approach to the design of a web-based data integration system using XML technologies. It includes some chapters covering practical solutions, based on XML and XML Query, for real world scenarios. However, despite of it is strongly related to the previous chapters, the focus of this work is not the generation of schema mappings or the interaction with ontologies, but the problem of distributing the queries over restricted autonomous interfaces. This is the reason why the two contributions have been placed in separate parts.

Chapter 2

Background Information

Terms as 'information', 'data', 'metadata', 'information retrieval', 'data retrieval', 'data integration', 'Semantic Web' and others, are specially susceptible of being interpreted in very different ways. So, to avoid misunderstandings, in this section I'm going to clarify, or at least narrow, the semantics of some important concepts related to this work.

2.1 Information vs. Data

Despite of in some works the terms 'information' and 'data' are used indistinctly, here I'm going to specially strict in separating the two concepts. According to The Free Online Dictionary of Computing [111], data are *"numbers, characters, images, or other method of recording, in a form which can be assessed by a human or (especially) input into a computer, stored and processed there, or transmitted on some digital channel. Computers nearly always represent data in binary. Data on its own has no meaning, only when interpreted by some kind of data processing system does it take on meaning and become information."*.

So, it is clear that is easier to define data than information, which requires a higher effort of abstraction. Now we can take profit that we've already separated these two concepts to locate another one, knowledge. According to [21] *"We had two decades which focused solely on data processing, followed by two decades focusing on information technology, and now that has shifted to knowledge. There's a clear difference between data, information, and knowledge. Information is about taking data and putting it into a meaningful pattern. Knowledge is the ability to use that information."*.

2.2 Metadata

In short, metadata is *"data about data"*. According to [9] metadata is *"information on the organization of the data, the various data domains, and the relationship between them"*. Tim Berners-Lee gives a more Web-centric definition, *"Metadata is machine understandable information about web resources or other things"*. We can difference two kinds of metadata [100], Descriptive Metadata and Semantic Metadata. Descriptive Metadata is external to the meaning of the data it describes, and pertains more to how it was created. For example, the Dublin Core Metadata Element Set [28] proposes 15 fields to describe a document. Semantic Metadata features the subject matter that can be found within the data it is describing. An example of semantic metadata could be some MPEG-7 [101] descriptors, that allow to describe for example that a video includes a football match in a rainy day. There are a lot of different uses of metadata, like cataloguing, content rating, intellectual property rights, etc. but nowadays some point to more ambitious uses (overall of semantic metadata) like for example the Semantic Web initiative.

2.3 Information Retrieval vs. Data Retrieval

According to [9], information retrieval *"deals with the representation, storage, organization of, and access to information items"*. The aim of an IR system is to facilitate the user access to the information in which he is interested. However, the *user information need* cannot be easily formalized, an it must be translated into a query processable by the retrieval system. Given the user query, the IR system aims to retrieve information which might be relevant to the user.

On the other hand, a data retrieval system aims to determine which objects of a collection satisfy clearly defined conditions as those in a relational algebra expression. For a data retrieval system, like a database, a single erroneous object among a thousands retrieved means a total failure. So, data retrieval deals with well defined data models, expressive query languages and performance issues, while information retrieval faces the problem of interpreting the contents of the information items to decide their relevance. The two concepts are not isolated, data retrieval is ever an important part of an IR system, and can be seen as a lower-level layer. Because this research work focuses in metadata, instead of on other traditional IR issues, it is in some way more related to data retrieval rather than to information retrieval. However, because the context keeps being IR systems for the Web, the references to IR aspects will be usual.

2.4 Traditional Information Retrieval vs. Multimedia Information Retrieval

Traditional information retrieval only deals with unstructured textual data. Traditional IR is an old discipline, with published books from even the last 70s like e.g. [131], and already classic conferences like ACM SIGIR (International Conference on Information Retrieval) or TREC (Text REtrieval Conference). Its research is based in solid and well characterized models, like the boolean model, the vector model or the probabilistic IR model.

The irruption of the Web and Web search engines has put IR at the *"center of the stage"* since the 90s. However the new context introduces new challenges for IR like the retrieval of heterogeneous multimedia contents. Multimedia data is rapidly growing in the Internet, and also metadata related to multimedia information objects, that of course include textual documents. Multimedia IR systems must support different kinds of media with very heterogeneous characteristics such as text, still and moving images, graphs and sound. This poses several interesting challenges, due to the heterogeneity of data and the fuzziness of information. Multimedia IR systems have an interesting feature for our concerns, they must handle metadata, because it is crucial for data retrieval, whereas traditional IR systems do not have such requirement.

2.5 Data Integration

The Data integration (also named Information Integration) research discipline study mechanisms for a seamless access to autonomous and heterogeneous information sources. These sources can vary from legacy databases to Semantic Web or P2P applications. Traditionally the target of a data integration system is to provide a mediation architecture in which a user poses a query to a mediator that retrieves data from underlying sources to answer the query. The constraints of the sources access, and their potentially different data models and schemas, are the challenges of a data integration system.

Data integration is the main topic of this thesis, and its evolution and current situation will be described in the *State of the Art and Problem Statement Chapter*.

2.6 Distributed Information Retrieval vs. Data Integration

Search engines for the Web or other information retrieval systems are usually based, according to [24], in a *single database* model of text retrieval, in which documents are copied to a centralized database, where they are indexed and made searchable (this model can be seen as the information retrieval version of data warehousing for data retrieval). However, some information is not accessible under this model (it can be queried but cannot be copied to the centralized database) for different reasons (size, volalility, interface restrictions). The alternative is a *multi-database* model, in which the central site (or any peer in a distributed peer-to-peer context), instead of storing copies of the documents, translates the user information need into queries to the different sources. This kind of model is studied by the Distributed Information Retrieval discipline, and has been also informally known as metasearch.

A distributed information retrieval system covers traditionally the following stages:

- *Source description*: The contents of each text database must be described
- *Source selection*: Given the descriptors and the user information need, which sources must be queried.
- *Source querying*: Map the information need to the selected sources and query them.
- *Results merging*: Merge the ranked lists returned by the different sources.

In some aspects data integration and distributed information retrieval are equivalent, and in some contexts the words are mixed. However, while distributed information retrieval targets to satisfy a user information need over unstructured data or semi-structured data sources, a data integration system aims to satisfy a query over also autonomous and heterogeneous, but structured data sources.

2.7 Metasearch

Metasearch is the informal name that refers to Web-based distributed information retrieval systems (see [33] or [135] for a more detailed definition). Because metasearch systems are supposed to solve some of the problems described in the previous section, and because they never have really gained the favour of Web users, some interesting conclusions can be extracted from their evolution.

In theory, the strength of a metasearch engine is its recall value, because it queries a set of conventional crawler-based search engines that cover certain subsets of the public indexable Web. However, some studies have demonstrated that these subsets, far from being disjoints, are highly overlapped [88]. Furthermore, recall is not precisely the main problem of conventional Web search systems, but precision. Another important drawback of metasearch systems is that their work often does neither rely over an agreement with their underlying sources nor over specific interchange protocols. This forces them to face the problem to manage query forms and results pages designed for human consumption and written in HTML. This is usually solved with hand-coded screen-scraping rules or similar things that reduce speed and difficult maintainability. Some metasearch defenders claim that these systems have an important advantage, the ability to access the hidden web, because they capture the results of the target sources on-the-fly, that allows them to harvest dynamically generated content. However, this style of doing things, without considering legal issues[1], is far from being elegant, remember the sentence of Tim Berners-Lee [14]:

"And so you have one program which is turning it from data into documents, and another program which is taking the document and trying to figure out where in that mass of glowing flashing things is the price of the book. It picks it out from the third row of the second column of the third table in the page. And then when something changes suddenly you get the ISBN number instead of the price of a book and you have a problem. This process is called 'screen scraping', and is clearly ridiculous."

2.8 Datalog

Chapters 3 and 4 make use of the Datalog language [43] in some examples related to initial data integration approaches. Datalog is an old (1978) database query language that syntactically is a subset of Prolog. Its logic basis made it popular in academic database research, but despite of its advantages over standard query languages like SQL it never succeeded in becoming part of commercial systems.

A Datalog query program consists of a finite set of Horn clauses $C_1, ..., C_k$ (the rules of the Datalog program). Horn clauses express a subset of statements of first-order logic with at most one positive literal:

$$L \Leftarrow L_1, ..., L_n$$

[1]Some conventional search engines explicitly forbid metasearch. In Google's terms of service page we can find "You may not send automated queries of any sort to Google's system without express permission in advance from Google".

Equivalent to:

$$L \vee \neg L_1, ..., \neg L_n$$

In Datalog each literal corresponds to an atomic formula $p(A_1, A_2, ..., A_n)$ where p is the relation name and A_i are variables or constants (names that start with an upper case letter are variables). Let's see a relation table written in Datalog:

```
employees(name, dept, id)
```

Let's see an example query program with one Datalog rule:

```
financeEmployees(X, Y) :- users(X, "finance", Y)
```

There are two types of relations: (1) base relations (physically stored in the database) and (2) derived relations (temporary relations that hold intermediate results). The general form of a rule is as follows:

$$p(x_1, x_2, ..., x_n) : -q_1(x_{11}, x_{12}, ..., x_{1m}), ..., q_k(x_{k1}, ..., x_{kp}), e.$$

Where q_i are base or derived relation names, e is an arithmetic predicate (any number) and each x_i appearing in p appears in at least one of the q_i's. The Datalog rule can be interpreted as:

$p(...)$ is true if $q_1(...)$ and $q_2(...)$ and ... $q_k(...)$ and e is true.

In Datalog an answer to an atomic query is a set of constants that satisfy the query. Answers are computed by using top-down (or backward-chaining) or top-down algorithms.

2.9 The Extensible Markup Language (XML)

According to [9] Markup is defined as *"extra textual syntax that can be used to describe formatting, actions, structure information, text semantics, attributes, etc."*. One example of markup can be the formatting commands of the popular text formatting software TeX.

In the late seventies was defined the Standard Generalized Markup Language (SGML), a metalanguage for tagging text developed by Charles F. Goldfarb and his group, and based on a previous work done at IBM. In 1996 the SGML Editorial Review Board became the XML Working Group under the auspices of the World Wide Web Consortium (W3C), chaired by Jon Bosak of Sun Microsystems and with the intermediation of Dan Connolly. This group developed The Extensible Markup Language [151] (XML), a subset of SGML which goal is to enable generic SGML *"to be served, received, and processed on the Web in the way that is now possible with HTML(also based on SGML). XML has been designed for ease of implementation and for interoperability with both SGML and HTML"*[151].

XML, the same as SGML, is not exactly a markup language, it is a metalanguage that can be used to define specific markup languages (like XHTML, MathML, SVG, etc.). That means that XML allows users to define new tags and structures for their own languages.

For some reasons, some obvious and other that will remain a mystery, XML have reached an amazing success worldwide. In an interconnected and global society, the interchange of data over a standard syntax has become a key issue, and here is where XML fits perfectly.

2.10 The Semantic Web

The Semantic Web is a promising initiative lead by the W3C which aim is to provide a data model for the Web, allowing information to be *understood* and processed also by machines. The oficial definition of the W3C [142] says *"The Semantic Web is the representation of data on the World Wide Web. [....] It is based on the Resource Description Framework (RDF), which integrates a variety of applications using XML for syntax and URIs for naming."*. So, it is clear that this initiative strongly relies on RDF, but what is the meaning of *"the representation of data"*? Often in this document we have referred to de difficulties related to search and retrieve information on the Web. One of the main reasons is the fact that the most part of Web data, despite of being processed by machines, can be only understand by humans. This include natural language text, still/moving images, audio, etc.

Before we have discussed the difference between information retrieval and data retrieval, saying that while data retrieval is appropriate for databases it is not appropriate (or not enough) for the Web. The reason is that the information on the Web, contrary to databases, does not have an underlying data model. So, *"the representation of data"* means two things, the development of a data model for the Web, and the dissemination of machine-understandable metadata (under the framework of the data model) linked in some way to the Web information. Another classic definition is that by Berners-Lee et al. [17] *"The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation."*.

2.11 Querying the Semantic Web

The Semantic Web initiative has opened a broad spectrum of opportunities for improving the search and retrieve of information on the Internet. Of course this is not casual, but one of the main targets of this new scenario as pointed in [15] or [12]. However, the consolidation of a standardised way to interchange semantic information is just another step in the race for interoperability. Other battles are being fight to rationalise the way this information is processed and search and retrieval are maybe the most important elements of the information feed chain. The challenge is to find efficient and rational ways to exploit this new information that begins to be disseminated over the net, and that, despite of it is formalised in a standard way (RDF [22]), it can be stored in different ways (embedded on HTML pages, in a database, in specific knowledge repositories, etc.) and it remains highly heterogeneous (an innumerable an unrestricted number of ontologies, potentially overlapped, can co-live in the Semantic Web).

This two key issues, how to locate and access the information, and how to manage heterogeneity, are of relevance for our analysis and also very related with what we have said in the previous sections. Some research works reflect special approaches to this, like the

Edutella project [102] that constitutes a distributed search network for educational resources and is based on P2P networking (its JXTA implementation [80]) and RDF. This interesting work uses the query exchange language family RDF-QEL-i (based on Datalog semantics and subsets) as standardised query exchange language format. Because Edutella peers are highly heterogeneous and have different kinds of local storage for RDF triples, as well as some kind of local query language (e.g., SQL) to enable them to participate in the network, wrappers are used to translate queries and results from the peer and vice versa.

Another work is Sesame [73], an extensible architecture implementing a persistent RDF store and a query engine capable to process RQL queries [3] [42]. Of special interest for us is TAP [49] a system that implements a general query interface called GetData, Semantic Negotiation and Web of Trust enabled registries. It introduces the concept of Semantic Search and describes an implemented system which uses the data from the Semantic Web to improve traditional search results. The GetData interface is a simple query interface to network accessible data presented as directed labelled graphs, in contrast to expressive query languages like SQL, RQL or DQL. This work defends deployability against query expressiveness.

Related to this project, and also with the query language of Edutella, is RDF-QBE [130], a mechanism for specifying RDF subgraphs, which they call 'Query by Example', that could allow a high performance standardised interface for retrieval of semantic information from remote servers. From all this study cases we can observe the latent necessity of defining a low-barrier mechanism that allow to harvest heterogeneous semantic information and how it generates a trade-off between deployability and expressiveness. Some of them (e.g. TAP) point the necessity to consider also other conventional or not-semantic search strategies, like crawler-based engines, when thinking in future applications.

2.12 Semantic Integration

The semantic integration research area is a joint effort between the people from databases and data integration, and the people from knowledge management and ontology research. It can be seen as a reformulation of some old problems like matching database schemas or answering queries using multiple sources. Ontologies can be the solution to some of these old challenges but also the source of new problems, like ontology alignment. This last topic tries to determine which concepts and properties represent similar notions between two ontologies and is one of the main goals in this area. Related to ontology alignment arise other questions, like how do we represent the mappings or what do we do with them. [106] presents a recent survey on semantic integration.

Semantic integration is one of the main topics of this thesis, and it will be discussed deeper in the *State of the Art and Problem Statement Chapter*.

2.13 Resource Description Framework (RDF)

According to [124] the Resource Description Framework (RDF) is *"a foundation for processing metadata; it provides interoperability between applications that exchange machine-*

understandable information on the Web". RDF is the main building block of the Semantic Web, a framework composed by some different but strongly related elements (a data model, a syntax and a subclassing language among other things). First RDF is a syntax-independent data model designed for representing named properties and property values. The basic model consists of three object types. On one hand *Resources*, as all things being described, like Web pages, images, videos or any other thing, even it is not digital (like a real-life book). The only requirement is that they have a name, and this name conforms to the URI [16] syntax. On the other hand we have *Properties*, as specific characteristics used to describe a resource. The value of a property (the object) can be another resource (specified by a URI) or it can be a literal (i.e. a simple string). Finally, a specific resource together with a named property plus the object (the value of the property for that resource) is a *Statement*, the third basic object type of the model.

RDF is being used in a variety of application areas, like in resource discovery, in cataloguing (for describing the content and content relationships of some information object), by intelligent software agents, in content rating, or in describing intellectual property rights for example. The combination of RDF with digital signatures aims to allow what is known as the "Web of Trust" [83]. The conceptual model of RDF is complemented with an XML interchange syntax. The syntax is needed to ensure the required interoperability when creating and exchanging metadata.

To complete the framework, RDF have a class system much like many object-oriented modelling systems. A collection of RDF classes is called a *schema*. A schema contains classes organised in a hierarchy, offering extensibility through subclass refinement. RDF schemas allow reusability of metadata definitions. The schemas themselves may be written in RDF, with the RDF Schema language [126]. RDF schemas are being used nowadays to serialise *ontologies*.

2.14 Ontology Web Language (OWL)

According to [112] the Ontology Web Language (OWL) is a language *" intended to be used when the information contained in documents needs to be processed by applications, as opposed to situations where the content only needs to be presented to humans. OWL can be used to explicitly represent the meaning of terms in vocabularies and the relationships between those terms. This representation of terms and their interrelationships is called an ontology "*

OWL is a vocabulary for describing properties and classes of RDF resources, complementing the RDFS capabilities in providing semantics for generalization-hierarchies of such properties and classes. OWL enriches the RDFS vocabulary by adding, among others, relations between classes (e.g. disjointness), cardinality (e.g. "exactly one"), equality, richer typing of properties, characteristics of properties (e.g. symmetry), and enumerated classes. The language has three increasingly expressive sublanguages designed for different uses:

- OWL Lite has the lowest formal complexity, and serves for simple classification hierarchies. It has some restrictions like e.g. permitting only cardinality values of 0 or 1.

- OWL DL tries to offer the maximum expressiveness while retaining computational completeness (all conclusions are guaranteed to be computable) and decidability (all computations will finish in finite time). OWL DL is so named due to its correspondence with description logics.
- OWL Full does not give any computational guarantee but offers the maximum expressiveness and the syntactic freedom (e.g. a class can be treated simultaneously as a collection of individuals and as an individual).

In this work I will focus always in OWL DL, even when I do not mention it explicitly.

2.15 OWL and Description Logics

OWL has the influence of more than 10 years of Description Logic research. It is supposed that this knowledge has served to choose the constructors and axioms supported carefully, balancing expressiveness and efficiency. This balance was achieved by basing OWL on the $\mathcal{SH}$ family of Description Logics [63]. Members of the $\mathcal{SH}$ family include the $\mathcal{SHIQ}$ Description Logic [62] and the $\mathcal{SHOQ}$ Description Logic [61], that overcomes some limitations of $\mathcal{SHIQ}$ by taking the logic $\mathcal{SHQ}$ and extending it with individuals and concrete datatypes.

The OWL Lite and OWL DL species are syntactical variants of Description Logic languages. OWL Lite can be seen as a variant of the $\mathcal{SHIF}(D)$ description logic language, which is itself just $\mathcal{SHOIN}(D)$ without the *oneOf* constructor and with the *atleast* and *atmost* constructors limited to 0 and 1 [60].

OWL DL is a variant of the $\mathcal{SHOIN}(D)$ language, which is itself an extension of the $\mathcal{SHOQ}(D)$ (adding inverse roles and restricted to unqualified number restrictions) [59]. OWL Full extends both OWL DL and RDF(S) and thus cannot be translated into a Description Logic language. Entailment in OWL Full is undecidable in the general case, because it allows arbitrary roles in number restrictions, which makes the logic undecidable [62].

Part I

State of the Art and Problem Statement

Chapter 3

State of the Art in Data Integration

The integration of data from multiple heterogeneous sources is an old and well-known research problem. It has been traditionally faced by the database and AI research communities, but the promise of integrating data on the WWW or Peer-to-Peer systems and its related technologies (XML, RDF, etc.) has attracted efforts from other disciplines. Below I explain the progress of this topic and what challenges lie ahead.

3.1 Historical Progress

The integration of multiple information sources aims at giving users and applications the illusion of interacting with one single information system. There are two general approaches to this problem, materialized integration and virtual integration.

Materialized integration is strongly related to materialized views in databases, and consists on first storing all data from all sources locally and then querying them. Data warehousing is a well-know example of materialized integration. It is suited for situations when data changes infrequently and a fast evaluation of complex queries is required. However it is not always possible or convenient to replicate and update all data from a set of sources. There are situations when the size or volatility of data (or the limitations imposed by the sources query interfaces) makes materialization impossible. This is the reason why virtual integration has become of increasing interest in recent years as it has matured.

Virtual integration aims to offer the same results without the constraint of having to store and update all data from all sources. In pure virtual integration the global schema is strictly a logical entity. Queries issued over it are dynamically rewritten at runtime and redirected to the underlying data sources. Resulting data is fetched from the sources through *wrappers* and merged. Here we are just interested in virtual integration, or simply data integration from now. There are several problems related to data integration, but the main ones are: 1) The ability to present an integrated (mediated) schema for the user to query, or the *modelling problem*, 2) The ability to reformulate the query to combine information from the different sources according to their relationships with the mediated schema, or the *querying problem*, and 3) The ability to efficiently execute the query over the various local and remote data sources.

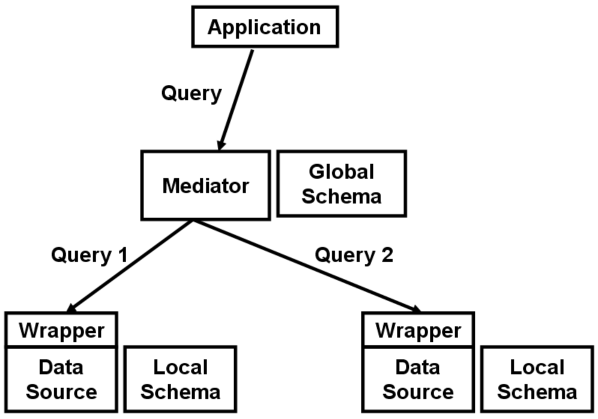


Figure 3.1: Architecture of a data integration system

3.1.1 Mediated Schema (the modeling problem)

We talk about a *mediated schema* when a data integration system employs a logical schema in order for several autonomous sources to interoperate (see fig. 3.1). This kind of schema is usually accompanied by the definition of *semantic mappings* (translations) between the mediated schema and the schemas of the different sources. This correspondence will determine how the queries posed to the system are answered.

The two classic approaches concerning mediated schemas and mappings modelling are the *global-as-view* and the *local-as-view*. The *global-as-view* approach or GAV [2](TSIMMIS)[26][50], consists on a mediated schema (the global schema) which is defined as a set of views over the data sources. This kind of mediation has the advantage that the user query can be simply merged with the view definitions (unfolded) obtaining a full query. The disadvantage of this approach is that the mediated schema is strongly coupled with the underlying source schemas and their changes, making it a bad solution for the Web context, where sources are autonomous and volatile.

The *local-as-view* approach or LAV [98][56][34] takes the inverse point-of-view and describes sources as views over the mediated schema. It has the advantage that changes on the underlying sources does not imply changes on the mediated schema. The disadvantage of this approach is the difficult to map the user query, referred to the mediated schema, to the different data sources. It's worth mention also the hybrid combination of GAV and LAV into the GLAV formalism [41].

3.1.2 Formalisation of the modelling problem

A formalisation of the *modeling problem* borrowed from [93] can be:

Definition 3.1.1. A data integration system $\mathcal{I}$ is a triple $<\mathcal{G}, \mathcal{S}, \mathcal{M}>$ where:

- $\mathcal{G}$ is the global schema (structure and constraints),
- $\mathcal{S}$ is the source schema (structures and constraints), and

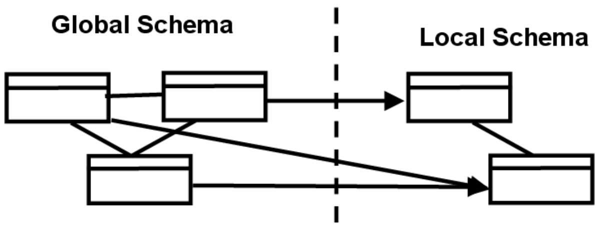


Figure 3.2: Global-as-view (GAV) approach

- $\mathcal{M}$ is the mapping between $\mathcal{G}$ and $\mathcal{S}$.

To specify the semantics of $\mathcal{I}$ we have to start with a source database $\mathcal{D}$ (source data coherent with $\mathcal{S}$). We call *global database* for $\mathcal{I}$ any database for $\mathcal{G}$. A global database $\mathcal{B}$ for $\mathcal{I}$ is said to be legal with respect to $\mathcal{D}$ if:

- $\mathcal{B}$ is legal with respect to $\mathcal{G}$, i.e., $\mathcal{B}$ satisfies all the constraints of $\mathcal{G}$;
- $\mathcal{B}$ satisfies the mapping $\mathcal{M}$ with respect to $\mathcal{D}$

We can also specify the semantics of queries posed to a data integration system. If q is a query of arity n and $\mathcal{DB}$ is a database, we denote with q^{DB} the set of tuples (of arity n) in $\mathcal{DB}$ that satisfy q. Given a source database $\mathcal{D}$ for $\mathcal{I}$, the answer $q^{I,D}$ to a query q in $\mathcal{I}$ with respect to $\mathcal{D}$, is the set of tuples t such that $t \in q^{B}$ for every global database $\mathcal{B}$ that is legal for $\mathcal{I}$ with respect to $\mathcal{D}$. The set $q^{I,D}$ is called *the set of certain answers* to q in $\mathcal{I}$ with respect to $\mathcal{D}$.

3.1.3 Formalisation of global-as-view (GAV) approach

When modeling with GAV, the mapping $\mathcal{M}$ associates to each element g in $\mathcal{G}$ a query q_S over $\mathcal{S}$.

Definition 3.1.2. A GAV mapping is a set of assertions, one for each element g of $\mathcal{G}$, of the form $g \rightsquigarrow q_S$

The idea is that each element g of the global schema should be characterized in terms of a view q_S over the sources. The mapping is explicitly telling the system how to retrieve data related to each element from the global schema. In this sense the GAV approach helps enormously the query processing design, but is just effective when the system is based on a set of sources that is stable.

Example 3.1.1. A data integration system over two sources of movies information could present the following global schema:

Global schema:

movie(Title, Year, Director)
european(Director)

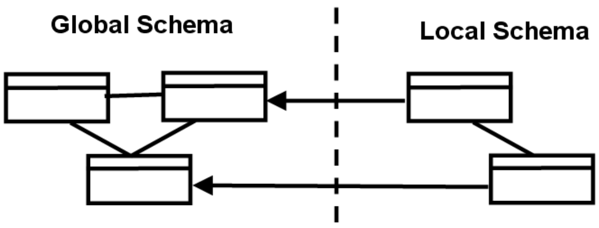


Figure 3.3: Local-as-view (LAV) approach

review(Title, Critique)

The two data sources could present the following local schemas:

Source 1: r1(Title, Year, Director) since 1960, european directors
Source 2: r2(Title, Critique) since 1990

Each entity of the global schema has assotiated one or more views over the sources:

$movie(\mathcal{T},\mathcal{Y},\mathcal{D}) \rightsquigarrow \{(\mathcal{T},\mathcal{Y},\mathcal{D})|r_1(\mathcal{T},\mathcal{Y},\mathcal{D})\}$
$european(\mathcal{D}) \rightsquigarrow \{(\mathcal{D})|r_1(\mathcal{T},\mathcal{Y},\mathcal{D})\}$
$review(\mathcal{T},\mathcal{R}) \rightsquigarrow \{(\mathcal{T},\mathcal{R})|r_2(\mathcal{T},\mathcal{R})\}$

3.1.4 Formalisation of local-as-view (LAV) approach

When modeling with LAV, the mapping $\mathcal{M}$ associates to each element s of the source schema $\mathcal{S}$ a query q_G over $\mathcal{G}$.

Definition 3.1.3. A LAV mapping is a set of assertions, one for each element s of $\mathcal{S}$, of the form $s \rightsquigarrow q_G$

The idea is that each source s should be characterized in terms of a view q_G over the global schema. This means that adding a new source just implies adding a new assertion in the mapping. This favours the maintainability and extensibility of the data integration system.

Example 3.1.2. The movies example under the LAV approach:

Global schema:

movie(Title, Year, Director)
european(Director)
review(Title, Critique)

In LAV the sources are featured as views over the global schema:

$r_1(\mathcal{T},\mathcal{Y},\mathcal{D}) \rightsquigarrow \{(\mathcal{T},\mathcal{Y},\mathcal{D})|movie(\mathcal{T},\mathcal{Y},\mathcal{D}) \wedge european(\mathcal{D}) \wedge \mathcal{Y} >= 1960\}$
$r_2(\mathcal{T},\mathcal{R}) \rightsquigarrow \{(\mathcal{T},\mathcal{Y},\mathcal{D})|movie(\mathcal{T},\mathcal{Y},\mathcal{D}) \wedge review(\mathcal{T},\mathcal{R}) \wedge \mathcal{Y} >= 1990\}$

Chapter 4

Query reformulation algorithms (the querying problem)

4.1 Query reformulation in LAV and GAV (the querying problem)

Strongly related to the mediated schema we found the algorithms for answering the user query in a data integration system. A initial query, targeting the logical mediated schema, must be translated into queries over the different data sources. In the case of the Global-as-view (GAV) approach, this problem reduces to *view unfolding* (unnesting). In the Local-as-view (LAV) approach, it translates to the more complex problem of *answering queries using views* [51], with some good solutions like MiniCon [119] or the bucket [56] algorithms.

The following two examples illustrates the querying problem for GAV and LAV:

Example 4.1.1. Querying the movies example under the GAV approach:

The query *"Title and critique of movies in 1998"* could be formalised (in respect to the global schema) as:

$\{(\mathcal{T},\mathcal{R})|movie(\mathcal{T},1998,\mathcal{D}) \wedge review(\mathcal{T},\mathcal{R})\}$

Because in GAV we have views for each schema entity, the query is processed by means of view unfolding, i.e., by expanding the atoms according to their definitions:

$movie(\mathcal{T},1998,\mathcal{D}) \rightarrow r_1(\mathcal{T},1998,\mathcal{D})$
$review(\mathcal{T},\mathcal{R}) \rightarrow r_2(\mathcal{T},\mathcal{R})$

Example 4.1.2. Querying the movies example under the LAV approach:

Having the same query of the previous example (*"Title and critique of movies in 1998"*) and its formalisation:

$\{(\mathcal{T},\mathcal{R})|movie(\mathcal{T},1998,\mathcal{D}) \wedge review(\mathcal{T},\mathcal{R})\}$

Because in LAV both the query and the mappings target the global schema, it is not trivial to determine how to map the query to the local sources. This process is performed by means of an inference mechanism that re-expresses atoms of the global schema in terms of atoms of the sources:

$\{(\mathcal{T},\mathcal{R})|r_2(\mathcal{T},\mathcal{R}) \wedge r_1(\mathcal{T},1998,\mathcal{D})\}$

While query reformulation looks easier in GAV, it is very complex (it needs reasoning) in LAV. However, LAV appears to be a better solution when autonomous and heterogeneous sources are present, like in the case of the Web. In such context we cannot rewrite the global schema and its mappings once and again, so we need a stable global schema and individual mappings that can be changed independently.

4.2 Answering queries using views

The first goal of the data integration system is to reformulate a user query $\mathcal{Q}$ to refer to the data sources. In the LAV approach the data covered by each source can be abstracted by a view $\mathcal{V}_i$ over the global schema. The first task of the system will be determining which views should be queried to achieve the best possible answer.

The old research paper (1995) *"Answering Queries Using Views"* [55] by A. Halevy et al. is probably the oldest and best known work facing the problem of determining the combination of data sources (modelled as views) that must be used to answer a given query in a LAV approach. This work considers the problem of rewriting a conjunctive query using a set of conjunctive views in the presence of a large number of candidate views.

As most part of similar works of these initial approaches it uses Datalog[1] to formalise the problem:

A conjunctive query $\mathcal{Q}$ has the form:

$q(X)$:- $e_1(X_1),...,e_n(X_n)$

where q and $e_1,...,e_n$ are predicate names. The atom $q(X)$ represents the answer relation and is called the *head* of the query. The atoms $e_1(X_1),...,e_n(X_n)$ are the *subgoals* of the query, where $e_1,...,e_n$ are database relations from the global schema.

4.2.1 Query containment

The query rewriting problem is closely related to the concept of *query containment.* We say that a query $\mathcal{Q}_1$ is *contained* in the query $\mathcal{Q}_2$, denoted by $\mathcal{Q}_1 \sqsubseteq \mathcal{Q}_2$, if the answer to $\mathcal{Q}_1$ is a subset of the answer to $\mathcal{Q}_2$. To determine if a conjunctive query $\mathcal{Q}_1$ is *contained* into another conjunctive query $\mathcal{Q}_2$ we must find for each subgoal of $\mathcal{Q}_2$ a subgoal in $\mathcal{Q}_1$

[1]see the *Background Information* chapters for a brief introduction to Datalog

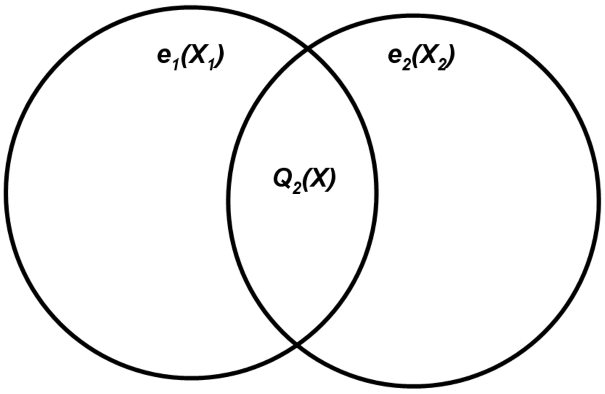


Figure 4.1: A conjunctive query $\mathcal{Q}_2$ with two subgoals

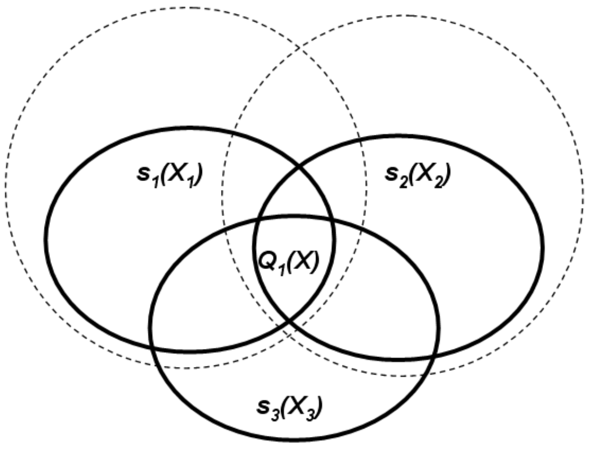


Figure 4.2: A conjunctive query $\mathcal{Q}_1$ with three subgoals that is contained in $\mathcal{Q}_2$

contained in it. Figures fig. 4.1 and fig. 4.2 shows graphically the concept of containment applied to conjunctive queries.

Example 4.2.1. A query $\mathcal{Q}_2$ that asks for people with blue eyes and blond hair contains a query $\mathcal{Q}_1$ that asks for women with blue eyes and blond hair because for each subgoal of $\mathcal{Q}_2$ (blue eyes, blond hair) $\mathcal{Q}_1$ has a subgoal contained in it (plus some other goals, like being a woman). In contrast, a query $\mathcal{Q}_3$ that asks for people with blue eyes is not contained in $\mathcal{Q}_2$ because it does not provide a subgoal contained in the subgoal "blond hair".

When all subgoals of a query contain subgoals of another query we call the set of containments a *containment mapping*. So we can say that a query $\mathcal{Q}_2$ contains $\mathcal{Q}_1$ if and only if there is a containment mapping from $\mathcal{Q}_2$ to $\mathcal{Q}_1$.

4.2.2 Rewriting a query using views

Given a conjunctive query $\mathcal{Q}$ and a set of views $\mathcal{V} = \mathcal{V}_1, ...\mathcal{V}_n$, we want to rewrite $\mathcal{Q}$ using just some of the views and comparison predicates.

Example 4.2.2. Consider the following global schema. The relation movie(m, n) stores the identifiers (m) of all movies and their names (n). The relation actor(a, n, y) stores identifiers (a) of actors, their names (n) and birth years (y). The relation starring(a, m) stores the relationship between movies and actors identifiers. The following query asks for the filmography of the actor 'Christopher Walken'.
Q(m) :- movie(m,n),actor(a,'Christopher Walken'),starring(a,m)

Database views are named queries that return a subset of the data in a database. They can also be modelled as conjunctive queries and formalised with Datalog rules. There are some differences between answering queries using real relational views and answering queries using virtual views representing data sources in a LAV-based data integration system. Two LAV views with the same definition are not assumed to contain the same tuples because they represent autonomous data sources. So, it makes sense to have the views:

V1(m) :- movies(m,n)

V2(m) :- movies(m,n)

V1 and V2 can represent two different movie databases containing different subsets of movies.

Example 4.2.3. Continuing with our example, consider the following views:
V1(m) :- movies(m,n)
V2(m) :- movies(m,n)
V3(a, m) :- starring(a,m)
V4(a) :- actors(a,n,y), y<1950

Now we can consider the problem of rewriting a query over a database using only views or comparison predicates (without directly using relation predicates).

Definition 4.2.1. (contained rewriting or simply rewriting) Let $\mathcal{Q}$ be a query, and $\mathcal{V} = \mathcal{V}_1, ...\mathcal{V}_n$ be a set of views. The query $\mathcal{Q}'$ is a rewriting of $\mathcal{Q}$ using $\mathcal{V}$ if $\mathcal{Q}' \sqsubseteq \mathcal{Q}$.

Example 4.2.4. One possible rewriting of the previous example query using the views is:
Q'(m) :- V1(m),V3(a,m),V4(a)

Another possible rewriting can be:
Q'(x,y) :- V2(m),V3(a,m),V4(a)

Example 4.2.5. Unfolding the views of the previous example we can see that the resulting rewritings are contained in the initial query, so they are valid rewritings. Take the first for example:

Q"(m) :- movie(m),actor(a,'spain'),starring(a,m),woman(a)

The obtained rewritings guarantee that results will not be outside the scope of the initial query, but they do not guarantee that they are the same results that could be obtained applying directly the query over a hypothetical local database containing all the data from the sources. This ideal situation is called equivalent rewriting and is pursued when answering queries using views is applied to query optimization and physical data independence in a local database.

Definition 4.2.2. (equivalent rewriting) Let $\mathcal{Q}$ be a query, and $\mathcal{V} = \mathcal{V}_1, ...\mathcal{V}_n$ be a set of views. The query $\mathcal{Q}'$ is an equivalent rewriting of $\mathcal{Q}$ using $\mathcal{V}$ if $\mathcal{Q}' \sqsubseteq \mathcal{Q}$ and $\mathcal{Q} \sqsubseteq \mathcal{Q}'$

In the context of data integration we pursue to obtain the biggest set of results possible using the given views. The best rewriting in this sense is called the maximally-contained rewriting.

Definition 4.2.3. (maximally-contained rewriting) $\mathcal{Q}'$ is a maximally-contained rewriting of $\mathcal{Q}$ using views $\mathcal{V} = \mathcal{V}_1, ...\mathcal{V}_n$ if (1) $\mathcal{Q}' \sqsubseteq \mathcal{Q}$ and (2) there is no other query $\mathcal{Q}''$ such that $\mathcal{Q}' \sqsubseteq \mathcal{Q}'' \sqsubseteq \mathcal{Q}$.

The maximally-contained rewriting of a conjunctive query can be obtained with the union of all possible contained rewritings.

Example 4.2.6. To obtain the maximally-contained rewriting of our example we simply perform the union of the two obtained rewritings. This is usually represented just by showing the list of rewritings:
Q'(m) :- V1(m),V3(a,m),V4(a) Q'(m) :- V2(m),V3(a,m),V4(a)

4.3 Parametrized views

The initial approaches to answering queries using views provide a formal basis for the data integration problem. However, data sources in the real world are difficult to be represented with a single view or with a finite set of views because they use to present parametrized query interfaces. These initial works on answering queries using views assume a finite set of views $\mathcal{V}$, but a parametrized query interface can be only represented by a potentially infinite number of views.

Example 4.3.1. Continuing with our example about movies, it is not realistic to assume that a data source can be represented with a finite view like:

V1(m) :- movies(m,n)

Probably the source would offer a query interface with some parameters, like e.g. the movie name. In this case instead of one view we would have one view for each possible name:

V1(m) :- movies(m,'Rio Bravo')
V2(m) :- movies(m,'Assault on Precinct 13')
...

To overcome this limitation some works have analysed the problem of answering queries using sources modelled by an infinite set of views. In [115] authors already considered this possibility, showing that it is important to be able to exploit the local processing power of sources to reduce the amount of data transmitted over the network. In this work is introduced the concept of a *parametrized view* as a conjunctive query that contains placeholders in argument positions.

A parametrized view $\mathcal{V}$ represents the set of all view definitions obtained by assigning a constant to each placeholder. Placeholders can be denoted by argument names beginning with an asterisk (*).

Example 4.3.2. We can rewrite the previous example with this parametrized view:
V1(m) :- movies(m, *n)

In [57] A. Halevy, A. Rajaraman and J. D. Ullman extend this idea showing that any infinite set of views can be partitioned into a finite set of equivalence classes, in such a way that all views in an equivalence class are also equivalent with respect to rewritings of a query $\mathcal{Q}$. The equivalence classes allow to keep applying the traditional algorithms of answering queries using views like [56].

4.4 Query processing

The resolution of a query in a data integration system can be divided in two stages, *query reformulation* and *query processing*. Query reformulation corresponds to the research around *answering queries using views*, and focus on the selection of the sources that can provide the best valid response to a given query. However, knowing which sources to query it is not enough. The obtained rewritten query of the first stage is a declarative query wich refers to the sources modelled as views. In a local system such a high-level query would be translated to a syntactic tree and then optimized for execution. By contrast, in a data integration system some of the algebraic operations can be performed locally at the sources, while others must be performed in the mediator. The query processing stage aims to generate the best execution plan for a given query and executing that plan with the help of the mediator and the wrappers of the sources.

Because the target systems are distributed, autonomous and heterogeneous, achieving a good performance can be a difficult task. In answer to this challenge, several works have considered *adaptive query processing* [71][72][7] where the systems starts with some execution plan and adapts it as the execution proceeds.

Chapter 5

Data Integration and XML

The classic data integration literature focused on the Relational Model for both queries and mappings till mid-90s. However, in late-90s researches turned their interest to a new and emerging data model, XML [68]. The new model aroused as a de-facto standard to expose and interchange data, so it was the ideal choose for systems pursuing data interoperability. Now, XML and its query languages are the selected interfaces for Web Services, XML-native databases and lots of other applications.

5.1 Mapping the classic data integration problems to XML

Integrating data from various XML sources arise the same problems described in the classic data integration literature, but new solutions need to be found to tackle the particularities of the new scenario.

The first of these classical problems is schema mapping. The schema in which terms is expressed the query (there's no need to call it the *Global Schema* if we are e.g. in a peer-to-peer context) must be someway mapped to the schema or schemas of the sources where the query will be actually executed. The simplest approach to such mapping is an attribute correspondence, where some property or attribute in one representation corresponds to some attribute in the other representation. We find an increased complexity when mapping concepts that are semantically the same, but the XML representations may be structured differently.

Example 5.1.1. This example, borrowed from [52], illustrates some of the problems of mapping XML schemas.

```
Source1.xml DTD:
pubs
  book*
    title
    author*
      name
    publisher*
      name
```

```
Source2.xml DTD:
authors
  author*
    full-name
    publication*
      title
      pub-type
```

The example shows how a simple schema describing books and authors can take different shapes. The difficult of obtaining a mapping between them will depend on the goal of that mapping. It may serve for simple migration tasks (translation of data from one schema to another), and then a simple translation template will be enough. However, it may be needed for querying purposes, and then a more complex strategy is needed, related to the old *query rewriting* problem described in previous sections.

5.2 XML query languages and data integration

XML query languages have been broadly used for the development of simple data integration applications. Mapping between schemas or defining wrappers with XSLT or XQuery can be a direct solution for some real world problems. These solutions generally are based on the manual coding of templates and updates, so they represent the modern version of the more primitive data integration approaches.

5.2.1 XSL Transformations (XSLT)

XSL Transformations (XSLT) is a language standardized by the W3C for transforming XML documents into other XML documents. XSLT is a component of the W3C's XML Stylesheet Language, and initially its main purpose was to be used in conjunction with a formatting language like XSL:FO, targetting the presentation layer independence. However, XSLT can be used independently, and it has been used in many application areas, but specially by the data integration community.

A transformation expressed in XSLT, called a *stylesheet*, describes rules for transforming a source XML document into a result XML document. An XSLT stylesheet associates *patterns* with *templates*. When a pattern is matched against an element in the source XML tree, the corresponding template is instantiated to generate XML code for the result document. This generation can include data from the source tree, but also can include new data.

The current version, XSLT 2.0 (W3C Candidate Recommendation 3 November 2005), is a revised version of the XSLT 1.0 Recommendation published on 16 November 1999. It is designed to be used in conjunction with XPath 2.0, which is defined in [69]. XSLT shares the same data model as XPath 2.0, which is defined in [152].

The capabilities of XSLT for transforming XML documents makes it a natural choice for data integration applications. In scenarios where heterogeneous XML schemas need to be mapped, XSLT stylesheets can be manually coded or semi-automatically generated to allow the conversion between the different schemas. XSLT has been used also for intra-model conversions, like RDF-to-XML. Another usage of XSLT has been in the definition of web wrappers. HTML code can be easily modified to become XHTML with tools like HTML Tidy [143], and then filtered with XSLT stylesheets. Lots of commercial products make use from XSLT data integration capabilities, like the Altova MapForce tool [99].

Example 5.2.1. This example shows how an input XML document can be transformed using an XSLT template. Take the followint XML document describing two movies.

```
intput.xml:

<?xml version="1.0"?>
<movies>
  <movie id="26">
    <title>Blade Runner</title>
    <year>1982</year>
  </movie>
  <movie username="27">
    <title>Rio Bravo</title>
    <year>1959</year>
  </movie>
</movies>
```

The following XSLT template is applied recursively to all the nodes of the underlying tree of the input document. The template translates the *movie* elements into *record* elements. It also translates the *id* attributes into equivalent elements.

```
template.xslt:
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:output method="xml" indent="yes"/>
 <xsl:template match="/">
    <transform>
       <xsl:apply-templates/>
    </transform>
 </xsl:template>
 <xsl:template match="movie">
     <record>
        <id>
           <xsl:value-of select="@id" />
        </id>
        <title>
           <xsl:value-of select="title" />
        </title>
     </record>
```

```
  </xsl:template>
</xsl:stylesheet>
```

This is the resulting XML document:

```
output.xml:
<?xml version="1.0" encoding="UTF-8"?>
<transform>
   <record>
      <id>26</id>
      <title>Blade Runner</title>
   </record>
   <record>
      <id>27</id>
      <title>Rio Bravo</title>
   </record>
</transform>
```

5.2.2 XML Query (XQuery)

The W3C's XML Query language (XQuery) [154] allows to query the logical structure of an XML document (defined in [152] in a SQL-like fashion. It is derived from the previous XML query language called Quilt, which in turn borrowed features from several other languages, including XPath 1.0, XQL, SQL, and OQL.

XQuery Version 1.0 is an extension of XPath Version 2.0. It enriches XPath functionality with FLWR expressions (FOR-LET-WHERE-SORT BY-RETURN), element constructors, variables, functions and updating capabilities.

There have been some discussion about the overlapping of XQuery and XSLT. This discussion can also take place in data integration scenarios. In principle XQuery and XSLT can be interchangeable on most part of situations, e.g. when mapping data from heterogeneous schemas or for the definition of wrappers. The final choose usually depends on the quality of tools and developers preferences. In general XSLT continues to be the primary choice for transforming XML data, while XQuery it is becoming the standard for querying and updating XML-based databases.

Example 5.2.2. This example shows how an input XML document can be transformed using an XQuery expression. The result of processing this query will be the same as in the previous example.

```
<transform>
{
  for $m in doc("input.xml")//movie
  return
    <record>
     <id>{ $m/@title/text() }   </id>
     <title>{ $m/title/text() } </title>
    </record>
}
</transform>
```

5.3 XML Data Integration Systems

The success of XML and its potential use in interoperability problems has fuelled lots of XML-related data integration projects like XQuare, Liquid Data (Enosys), Nimble, XML Information Workbench, Callixa, Metamatrix, Xyleme, Tukwila or Raccoon. Here I describe the features of three representative cases.

5.3.1 Tukwila

The Tukwila system, developed at the The University of Washington Database Research Group, is an example of the LAV approach applied to XML data. It uses the MiniCon [119] algorithm to reformulate the queries posed over the global schema into queries over the local XML sources. Tukwila is a native-XML integration system, because it operates directly over *non materialized* XML (not converted to another internal representation). The system introduces the X-scan operator, that allows to process XML data as it is being received (streaming XML).

Today Tukwila already is an old academic prototype, but it was designed by some active researchers in data integration like Zachary Ives and Alon Halevy, and provides advanced features not present in other commercial systems.

5.3.2 Enosys

The industrial success of a research approach do not always entail that it was the better choice, but at least demonstrates it was viable and well motivated. We can find such a success in an XML-based data integration system, the Enosys XML Integration Platform [116]. This system, based on the wrapper-mediator architecture, allows querying heterogeneous data sources abstracted with XML schemas. Wrappers (or XMLizers in the project's terminology) uses XML schemas as logical views of the sources, and a mediator resolves XQuery expressions over the sources.

On June 18, 2003, Enosys Software was acquired by BEA Systems, Inc. Now the system is part of the BEA's AquaLogic DSP, formerly known as Liquid Data, an XQuery-based Enterprise Information Integration (EII) solution that takes a data service layer-based approach to data integration.

5.3.3 XQuare Fusion

XQuare (XQuery Advanced Runtime Environment), previously known as XQuark, is a set of open source Java modules for extending J2EE platforms with XML-based, heterogeneous information integration capabilities. Instead of being an API, XQuare is designed to be embedded into Java-based Web or application servers, and rely on the standard J2EE services for exchanging, processing and publishing XML information. The goal of XQuare is presenting to applications a single, uniform XML view of the different data sources, which can then be queried with XQuery to produce XML documents. Accessible data sources include relational databases, XML documents, Web Services and any XQuery-enabled data source and JCA connectors. Last known release was in September 10, 2005.

Chapter 6

Semantic Integration

6.1 Ontologies and Data Integration

The Data Integration discipline has studied during almost two decades mechanisms to allow several autonomous data sources to interoperate. The main limitation of traditional data integration systems has been the impossibility to automatically establish semantic mappings between the data schemas of the different sources. Schemas obtained from traditional data models (e.g. relational and XML) are built with a reduced set of simple and meaningless constructs and lots of human readable labels. These kinds of data structures are designed to being interpreted just for humans, and automatically establishing semantic mappings among them becomes a very difficult and imprecise task.

The recent success of not so recent semantic-rich modelling languages under the global name of *The Semantic Web Initiative* has raised a new opportunity and challenge to the data integration community. Ontologies, instead of schemas, are the new way to represent information domains. They are built with a rich set of meaningful constructs provided by the Semantic Web modelling languages like RDFS [126] and OWL [112].

Because ontologies are built with standard semantic operators, a software agent could try to perform some automatic interpretations of the represented meaning. This could allow for example to automatically entail the semantic mappings between two different ontologies. However, the set of standard semantic operators are still very small, and today ontology modelling involves a lot of ambiguous natural-language labels. Obviously, as greater is the set of standard semantics of an ontology, easier will be its automatic processing by software agents. This is the reason of the proliferation of initiatives to standardize an *Upper Ontology*, like the IEEE SUMO (Suggested Upper Merged Ontology) [105], DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering) [44] or Cyc/OpenCyc [92].

While the fight for a standard semantic basis remains at the top level, at the domain level ontology standardization is not the goal. For many knowledge domains (anatomy, web directories, digital rights management, music, etc.), several overlapping ontologies have been engineered. Each is a different abstraction and representation of the same or similar concepts. To enable collaboration within and across information domains, software agents require the semantic alignment (mapping) of the different formalisms. It is the same old problem of *Schema Mapping* from Data Integration, but now with new promising expectatives and under the name of *Ontology Alignment*. This topic has attracted a lot of interest recently, even being object of an international contest, The Ontology Alignment Evaluation Initiative 2005 [108].

6.1.1 Semantic integration challenges

[106] enumerates the three main dimensions of semantic-integration research:

Mapping discovery (ontology alignment)

How do we define the similarity between two ontology entities. And, given two ontologies, how do we find the similarities between them.

Declarative formal representations of mappings

Once we have found the mappings between two ontologies, how do we represent this new knowledge to enable reasoning with mappings.

Reasoning with mappings

What do we do with the obtained mappings, how we use them to answer queries, how we face their uncertainty.

6.2 Ontology Alignment

Ontology alignment (or matching) is the operation that takes two ontologies and produces a mapping between elements of the two graphs that correspond semantically to each other. Several ontology alignment algorithms have been provided like PROMPT [107], GLUE [32], Ontrapro [6], OLA [37] or FOAM [36].

Definition 6.2.1. (from [35]) Given two ontologies $\mathcal{O}$ and $\mathcal{O}'$, an *alignment* between $\mathcal{O}$ and $\mathcal{O}'$ is a set of correspondences (i.e., 4-uples): $< e, e', r, n >$ with $e \in \mathcal{O}$ and $e' \in \mathcal{O}'$ being the two matched entities, r being a relationship holding between e and e', and n expressing the level of confidence [0..1] in this correspondence.

It is typically assumed that the two ontologies are described within the same knowledge representation language (e.g. OWL [112]). Here I will focus on automatic and autonomous alignment, but other semi-automatic and interactive approaches exist.

Example 6.2.1. Let's see simple example. Figures 6.1 and 6.2 present two ontologies, $\mathcal{O}_{\mathcal{A}}$ and $\mathcal{O}_{\mathcal{B}}$ respectively.

A possible alignment $A1$ (to simplify, the relation is always "=" and the confidence is always 1.0) is defined as follows:

```
<a:Human,b:People,=,1.0>
<a:Director,b:Manager,=,1.0>
<a:Staff,b:Employee,=,1.0>
<a:directs,b:supervise,=,1.0>
```

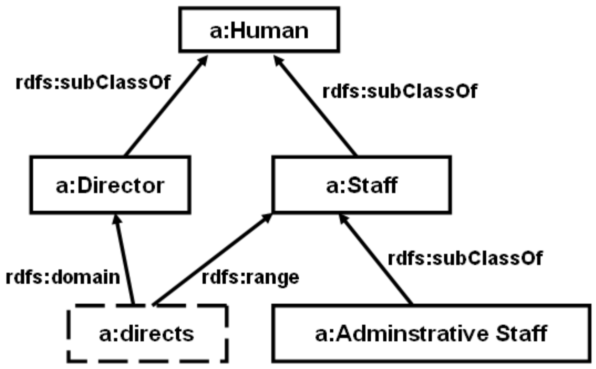


Figure 6.1: The RDF graph of $\mathcal{O}_{\mathcal{A}}$

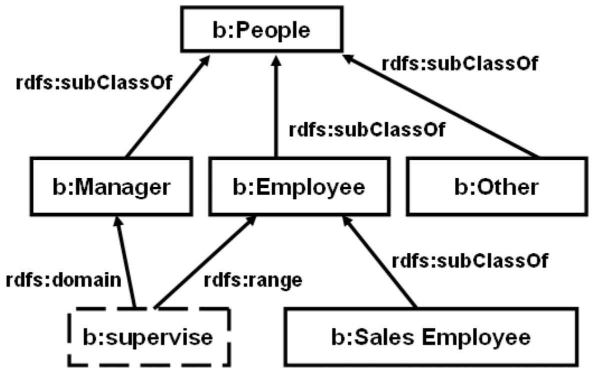


Figure 6.2: The RDF graph of $\mathcal{O}_{\mathcal{B}}$

Another reasonable alignment $A2$:

```
<a:Human,b:People,=,1.0>
<a:Director,b:Manager,=,1.0>
<a:Staff,b:Sales Employee,=,1.0>
<a:directs,b:supervise,=,1.0>
```

And an obviously wrong alignment A3:

```
<a:Human,b:Manager,=,1.0>
<a:Director,b:Employee,=,1.0>
<a:directs,b:Sales Employee,=,1.0>
```

6.2.1 Alignment Methods

The ontology alignment problem has an important background work in discrete mathematics for matching graphs [58][114], in databases for mapping schemas [122] and in machine learning for clustering structured objects [19]. It is closely related to the concept of *similarity*, an inverse measure of the distance between entities.

Most part of ontology alignment algorithms rely on some *semantic similarity* measure, used to deduce that two different data items correspond to the same information. Semantic similarity between ontology entities (within the same ontology or between two different ones) may be defined in many different ways. For example, it may be defined in terms of topological patterns. Given a pair of entities, c and c', a traditional method for measuring their similarity consists of calculating the distance between them in the graph. The shorter this distance, the higher the similarity. This is commonly known as the *edge counting method.*

Topological similarity methods have evolved from this simple idea, but there exist other similarity methods based e.g. in *information theory* . The recently celebrated Ontology Alignment Evaluation Initiative 2005 [108] has shown that best alignment algorithms combine different similarity measures.

6.2.2 Similarity measures

There are many different ways to define the similarity between ontologies. [37] provide a classification (updating [122]):

- **terminological (T)** comparing the labels of the entities; stringbased (TS) does the terminological matching through string structure dissimilarity (e.g., edit distance); terminological with lexicons (TL) does the terminological matching modulo the relationships found in a lexicon (i.e., considering synonyms as equivalent and hyponyms as subsumed);

- **internal structure comparison (I)** comparing the internal structure of entities (e.g., the value range or cardinality of their attributes);

- **external structure comparison (S)** comparing the relations of the entities with other entities; taxonomical structure (ST) comparing the position of the entities within a taxonomy; external structure comparison with cycles (SC) an external structure comparison robust to cycles;
- **extensional comparison (E)** comparing the known extension of entities, i.e. the set of other entities that are attached to them (in general instances of classes);
- **semantic comparison (M)** comparing the interpretations (or more exactly the models) of the entities.

This taxonomy is inherited from the study of similarity in relational schemas, and IMHO can be simplified to three categories when being applied to ontologies: **Lexical**, **Topological** and **Extensional**.

Lexical approaches

Lexical (or terminological) similarity is based in applying *information retrieval* techniques to match labels of entities. Labels are written in natural language and constitute one of the main sources of ambiguity in an ontology. However, all best algorithms of the Ontology Alignment Evaluation Initiative 2005 make use of lexical similarity measures at their first stages.

Topological (structural) approaches

The initial works around ontologies just focus on is-a constructs (taxonomies). The first ways to evaluate semantic similarity in a taxonomy were based only on the topology of the concept tree. Works like [121] and [90] measure the distance between the different nodes. The shorter the path from one node to another, the more similar they are. Given multiple paths, one takes the length of the shortest one. [148] finds the path length to the root node from the least common subsumer (LCS) of the two concepts, which is the most specific concept they share as an ancestor. This value is scaled by the sum of the path lengths from the individual concepts to the root. [89] finds the shortest path between two concepts, and scales that value by the maximum path length in the is–a hierarchy in which they occur.

However, the problem of this approach is that it relies on the notion that nodes in the taxonomy represent uniform distances. Actually, there can be a big variability in the distance covered by a single taxonomic node, specially when certain sub-taxonomies are much denser than others (e.g., biological categories).

Recently, new works like [20] define more sophisticated topological similarity measures, based on graph matching from discrete mathematics. These new graph-based measures suit the particularities of the new ontologies, built with more expressive languages like OWL [112]. Their use by some of the best alignment algorithms of the Ontology Alignment Evaluation Initiative 2005 (e.g. [64]) arises some expectation over this way of measuring the similarity of two concepts.

Extensional approaches

In some cases we know only the labelled structure of an ontology. However, in other situations we can also have some information about the instances corresponding to the classes and properties defined in the ontology (its extension). If these instances are enough representative, they can offer some relevant information to the similarity measurement. Extensional or corpusbased measures are related to statistics, Machine-Learning and Information Theory.

One of the oldest extensional measure is *res*, defined by Resnik in 1995 [129]. It was followed by two measures, *lin* [94] and *jcn* [79], that augment the information content of the LCS of two concepts with the sum of the information content of the individual concepts. [94] scales the information content of the LCS by this sum, while [79] subtracts the information content of the LCS from this sum.

More recent information-theoretic approaches are [30], [32] (GLUE) and [67]. They are essentially based in the concept of *joint probability distribution* defined in [38]. This distribution consists of the four probabilities: $P(A,B)$, $P(A,\overline{B})$, $P(\overline{A},B)$, and $P(\overline{A},\overline{B})$. A term such as P(A,B) is the probability that a randomly chosen instance from the universe belongs to A but not to B, and is computed as the fraction of the universe that belongs to A but not to B.

Practical uses of extensional information-theoretic similarity measures exist, like their application to Functional Genomics [8].

6.3 GMO. A structure-based semantic similarity algorithm

One of the best behaving algorithms of the Ontology Alignment Evaluation Initiative 2005 was *Falcon-AO*. Among other tools it makes use of a lexical similarity resolver and an interesting structural similarity strategy called GMO (Graph Matching for Ontologies [64]). GMO is interesting because it is a purely automatic algorithm for finding structural similarities between OWL-DL ontologies, and because it obtained excellent results in tests where lexical labels where obfuscated (to evaluate the behaviour of structural similarity strategies). Among other particularities, GMO simplifies the alignment of ontologies defined with rich modelling languages like OWL-DL because, instead of managing each relationship (is-a, part-of,...) specifically, it makes use of the underlying directed bipartite graph of the participating ontologies.

6.3.1 Graph similarity calculation algorithm

GMO is based on the structural similarity calculation described in [20], that is based on the following updating equation to compute the similarity matrix:

Definition 6.3.1. $X_{k+1} = BX_kA^T + B^TX_kA, k = 0, 1, ...$
where X_k is the $n_B * n_A$ similarity matrix of entries x_{ij} at iteration k, and A and B are the adjacency matrices of $\mathcal{G_A}$ and $\mathcal{G_B}$ respectively. [20] demonstrates that the normalized even and odd iterations of this equation converge.

Let's decompose this basic equation for a better understanding of its behaviour. Once we have a similarity matrix between $\mathcal{G}_{\mathcal{B}}$ and $\mathcal{G}_{\mathcal{A}}$, we can obtain the relationships between entities of $\mathcal{G}_{\mathcal{B}}$ and entities of $\mathcal{G}_{\mathcal{A}}$ by using the following formula:

$rel_{ba} = BX_k$

This new matrix describes the elements of $\mathcal{G}_{\mathcal{B}}$ in terms of their relationship with the elements of $\mathcal{G}_{\mathcal{A}}$. We can compare this matrix with A, that describes the elements of $\mathcal{G}_{\mathcal{A}}$ in terms of their relationship with themselves:

$sim_{ba} = rel_{ba}A^T$

The resulting matrix is already a similarity matrix between $\mathcal{G}_{\mathcal{B}}$ and $\mathcal{G}_{\mathcal{A}}$, but it describes only the relationship between $\mathcal{G}_{\mathcal{B}}$ and $\mathcal{G}_{\mathcal{A}}$ w.r.t. how elements of $\mathcal{G}_{\mathcal{B}}$ relate to elements of $\mathcal{G}_{\mathcal{A}}$. We must add the equivalent formulas and we obtain the final equation:

$X_{k+1} = BX_kA^T + B^TX_kA, k = 0, 1, ...$

That can be seen as:

$X_{k+1} = sim_{ba} + sim_{ab}$

Where:

$rel_{ba} = BX_k$

$sim_{ba} = rel_{ba}A^T$

$rel_{ab} = X_kA$

$sim_{ab} = B^Trel_{ab}$

Example 6.3.1. Let's see a simple example. Take the following trivial graphs $\mathcal{G}_{\mathcal{A}}$ and $\mathcal{G}_{\mathcal{B}}$.

Note that initially the similarity matrix X_0 is set to 1. If we start the process already knowing the similarity values of some pair of entities, we can modify this matrix accordingly, and keep the known values between iterations. Let's calculate the similarity between $\mathcal{G}_{\mathcal{A}}$ and $\mathcal{G}_{\mathcal{B}}$:

$$A = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} B = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} X_0 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

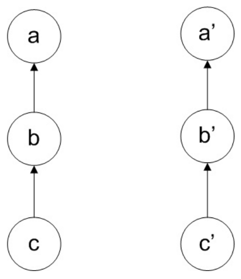


Figure 6.3: $\mathcal{G}_{\mathcal{A}}$ (left) and $\mathcal{G}_{\mathcal{B}}$ (right)

$$X_1 = BX_0A^T + B^TX_0A = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

$$X_1 = X_1/frobeniusNorm(X_1) = \begin{pmatrix} 0,316 & 0,316 & 0 \\ 0,316 & 0,632 & 0,316 \\ 0 & 0,316 & 0,316 \end{pmatrix}$$

Iterating the algorithm 22 times it converges to the following result:

$$X_{22} = \begin{pmatrix} 0,577 & 0 & 0 \\ 0 & 0,577 & 0 \\ 0 & 0 & 0,577 \end{pmatrix}$$

So, as expected the entities a', b' and c' (rows) are similar to a, b and c (columns) respectively.

6.3.2 GMO adaptation of the graph similarity algorithm to OWL-DL

GMO takes the graph structural similarity calculation of [20] and adapts it to OWL-DL ontologies.

Definition 6.3.2. (from [64]) Let $\mathcal{G}'_{\mathcal{A}}$ be the RDF directed labelled graph of $\mathcal{O}_{\mathcal{A}}$. The directed bipartite graph of ontology $\mathcal{O}_{\mathcal{A}}$, denoted by $\mathcal{G}_{\mathcal{A}}$, is a derivation of $\mathcal{G}'_{\mathcal{A}}$ by replacing the "s" (subject) edges with edges pointing to statement nodes, and the "p" (predicate) and "o" (object) edges with edges pointing from statement nodes. The adjacency matrix of $\mathcal{G}_{\mathcal{A}}$ is called the matrix representation of ontology $\mathcal{O}_{\mathcal{A}}$, denoted by $\mathcal{A}$.

Because of the different nature of the ontology entities (classes, properties, statements, shared entities, etc.) it is convenient to give the input matrices the following block structure,

$$A = \begin{pmatrix} 0 & 0 & A_{ES} \\ 0 & 0 & A_S \\ A_E & A_{OP} & 0 \end{pmatrix}$$

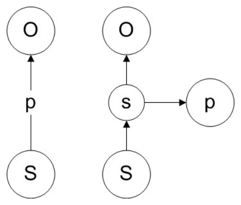


Figure 6.4: Comparison between an RDF graph (left) and its correspondant directed bipartite graph (right).

$$B = \begin{pmatrix} 0 & 0 & B_{ES} \\ 0 & 0 & B_S \\ B_E & B_{OP} & 0 \end{pmatrix}$$

- A_{ES} and B_{ES} represent the connections from external entities to statements in A and B respectively.

- A_S and B_S represent the connections from internal entities to statements in A and B respectively.

- A_E and B_E represent the connections from statements to external entities in A and B respectively.

- A_{OP} and B_{OP} represent the connections from statements to internal entities in A and B respectively.

External entities are usually those constructs defined by RDFS or OWL, built-in data types and literals.

As said before GMO uses the updating equation from [20]:

$X_{k+1} = BX_kA^T + B^TX_kA, k = 0, 1, ...$

The matrix X_k includes also the similarity related to statements and external entities. It can be decomposed as follows:

$$X_k = \begin{pmatrix} E_{BA} & 0 & 0 \\ 0 & O_k & 0 \\ 0 & 0 & S_k \end{pmatrix}$$

- E_{BA} represents the similarity among external entities.

- O_k represents the similarity among internal entities.
- S_k represents the similarity among statements.
- Other similarities are kept to zero (e.g. between statements and internal entities)

Initially $S_k = 1$, $O_k = 1$, and E_{BA} is set in advance as an identity matrix because external entities are supposed to be the same. Take for example only three external entities, *subClassOf*, *range* and *domain*. Their crossed similarity matrix would be:

$$E_{BA} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

For each iteration S_k and O_k are recalculated and normalized using the sum of the frobenius norm of the three matrices (E_{BA} is kept unchanged). Finally S_k and O_k are normalized again but with the 2-norm.
Some improvements of the algorithm described in [64] include a further classification of entities (in classes, properties and instances) that improves the scalability and performance.

6.3.3 Concept of similarity in GMO

The traditional definitions of structural similarity are usually based on the distance among nodes. This concept of similarity is inherited from the graph matching problem from discrete mathematics. However, in a knowledge representation scenario, the same or similar information can be represented taking a wide range of different shapes. So, simple graph-based similarity can arise totally arbitrary results. Intuitively, similarity of two concepts can be defined in terms of how these two concepts relate to the world they share. Two red objects are similar w.r.t. the colour dimension, but their similarity cannot be determined in a general way.

Because GMO is based in the directed bipartite graph of the participating ontologies, some modelling constructs like *subClassOf*, *range* or *domain* appear as shared external entities in the input graphs. Initially, GMO measures similarity of entities comparing the way they relate to these shared entities. As the algorithm iterates (structural similarity algorithms are always iterative), entities appearing to be similar can be also taken as a reference to find similarities between other entities [1]. This is a more rigorous and less arbitrary concept of similarity than those based on node distance. It compares how entities relate to common concepts, so it is closer to the human interpretation process, in which the meaning of something is entailed from how it relates to things for which the meaning is already known.

[1][20] demonstrates that the algorithm converges

6.3.4 An example

Example 6.3.2. Let's see a simple example. Take the following graphs $\mathcal{G}'_{\mathcal{A}}$ and $\mathcal{G}'_{\mathcal{B}}$.

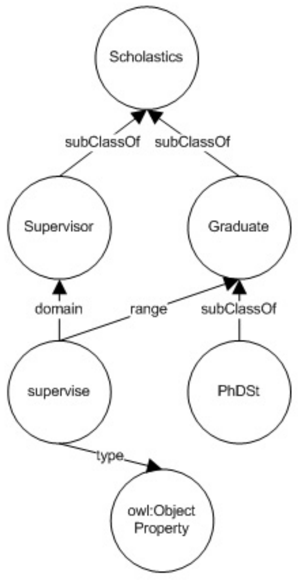


Figure 6.5: $\mathcal{G}'_{\mathcal{A}}$

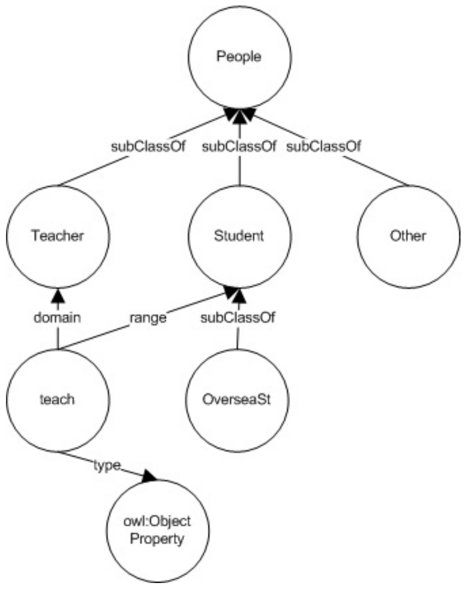


Figure 6.6: $\mathcal{G}'_{\mathcal{B}}$

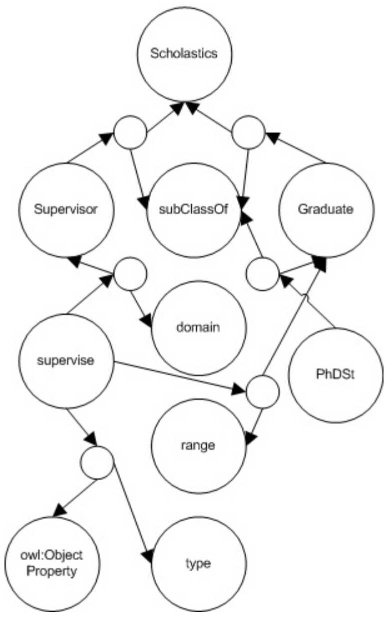


Figure 6.7: $\mathcal{G}_{\mathcal{A}}$

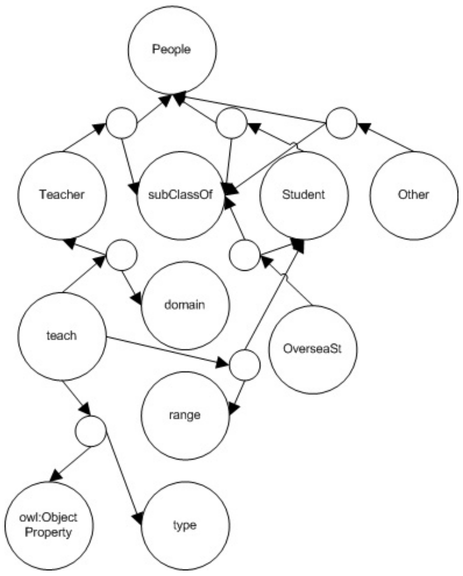


Figure 6.8: $\mathcal{G}_{\mathcal{B}}$

Iterating the algorithm 22 times it converges to the following result (Rows: b:Teacher, b:OverseaStudent, b:People, b:Other, b:Student; Columns: a:Graduate, a:Scholastics, a:PhdStudent, a:Supervisor):

$$X_{12} = \begin{pmatrix} 0,049 & 0 & 0,014 & 0,106 \\ 0,013 & 0 & 0,02 & 0,013 \\ 0,051 & 0,125 & 0 & 0 \\ 0,018 & 0 & 0,014 & 0,018 \\ 0,145 & 0,029 & 0,014 & 0,049 \end{pmatrix}$$

The similarity between b:teach and a:supervise is 0,446.
After normalization:

$$X_{12} = X_1/maxValue(X_1) = \begin{pmatrix} 0,336 & 0 & 0,098 & 0,73 \\ 0,09 & 0 & 0,134 & 0,09 \\ 0,353 & 0,863 & 0 & 0 \\ 0,127 & 0 & 0,098 & 0,127 \\ 1 & 0,201 & 0,098 & 0,336 \end{pmatrix}$$

The similarity between b:teach and a:supervise is 1.
So, as expected the entities a', b' and c' (rows) are similar to a, b and c (columns) respectively.

6.4 Upper Ontologies

Some recent and not so recent initiatives pursue to develop a general-purpose ontology (a.k.a. upper ontology), formalizing concepts such as processes and events, time and space, physical objects, and so on. These upper ontologies aim to offer some basic and standard meaning building blocks to allow domain-specific ontologies extend them.

As noted by [106], this scenario is different from the traditional data integration scenario, where a global schema (a common view on different local schemas) is usually generated once the underlying local schemas are already known. User queries are written in terms of the global schema, and the integration problem reduces to 1) Map the local schemas to the global schema (using the Global As View (GaV) or the Local As View (LaV) approaches) and 2) Answer the query using the defined mappings.

An upper ontology does not aim to be a view over all its derived domain-ontologies, nor pretending standard queries being written in its terms. An upper ontology is usually more general, since it define constructs for ontologies yet to be developed. However, it serves also to the data integration goals, sice increasing the number of standard semantics it also improves the confidence of Ontology Alignment algorithms.

Some upper or top-level ontologies are SUMO [105], DOLCE [44] and Cyc/OpenCyc [92]. Very related to this idea -despite it is not strictly an upper ontology- we find also WordNet [95].

6.4.1 IEEE SUMO

SUMO (Suggested Upper Merged Ontology) [105] is an effort by the IEEE Standard Upper Ontology Working Group aimed at developing *a standard upper ontology that will* *promote data interoperability, information search and retrieval, automated inferencing, and natural language processing.*. SUMO tries to standardize a hierarchy of some basic ground concepts like *Object*, *ContinousObject*, *Process*, *Quantity* or *Relation*.

6.4.2 DOLCE

DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering) [44] is a formal foundational ontology developed as an upper ontology in the WonderWeb project. DOLCE aims to provide a set of common semantics to achieve interoperability among ontologies related to WonderWeb. According to [44], it aims at capturing *ontological categories underlying natural language and human common-sense.*

6.4.3 WordNet

WordNet is an online lexical reference system, developed at Princeton University. English nouns, verbs, adjectives and adverbs are organized into synsets (synonym sets), each representing one underlying lexical concept that is semantically identical to each other. Despite of WordNet does not define itself as an ontology, synsets are cross-linked through relationships such as synonymy and antonymy, hypernymy and hyponymy (Subclass-Of and Superclass-Of) meronymy and holonymy (Part-Of and Has-a). So, we can consider WordNet as a special kind of upper ontology.

6.4.4 Cyc/OpenCyc

Doug Lenat's Cyc (from enCYClopedia) Project [92] was begun in 1984 as an attempt to build a *universal* expert system. The project resolved basic questions about representing time, substances, perception, etc., and the original emphasis on frames shifted towards first-order predicate calculus instead. The initial idea of a unified knowledgebase was replaced with the idea of many partially-independent *micro-theories.*

Cyc's main goal was constructing a foundation of basic *common sense* knowledge, a semantic substratum of terms, rules, and relations. It intended to provide a layer of meaning that can be used by other programs (such as domain-specific expert systems). Nowadays its open source version, OpenCyc, is still progressing, and contains over 47,000 concept terms and over 300,000 facts.

Chapter 7

Current Challenges in Data integration

Data integration has evolved in parallel to computer networks and computer paradigms. First data integration problems were related to the evolution of local area networks. The success of Internet and the WWW fed a new generation of problems, and solutions for some of them. Now XML, the Semantic Web, and the P2P paradigm arise new challenges for this discipline.

7.1 Semantic Mappings Generation: Schema matching and Ontology Alignment

Most part of approaches of data integration (GAV, LAV and GLAV) rely on the semantic mappings between a set of different data sources and a mediated schema. Traditionally these mappings have been written manually, being this the main drawback of data integration systems. Manual mapping generation is a costly and error-prone task, and -what is worst- it entails serious maintainability problems.

For the moment, complete automation of the generation of semantic mappings seems not to be possible. This task entails the complete understanding of the semantics of the source schemas or ontologies, that surpasses all known AI techniques. However, the topology and lexical information of the schemas and ontologies, or even their related data, provide clues that can serve to help the process of generation and maintainment as we have seen in the chapter about Semantic Integration. Now, the focus has turned to Ontology Alignment, because ontologies have potentially better possibilities for semantic integration than schemas. However, schema integration research has achieved good results these last years, like the semi-automatic approach in [122], or the work of A. Doan [31], whose Ph.D. thesis *"Learning to Match the Schemas of Databases: A Multistrategy Approach"* won the 2003 ACM Doctoral Dissertation Award.

7.2 Answering queries using ontology alignments

Data integration aims at giving users and applications the illusion of interacting with one single information system. This interaction usually takes the form of a query that the user issues to the system, which it must process and return a satisfactory answer. In ontology-based information systems, user queries are translated to queries or other retrieval tasks over the ABox defined[1].

One of the main goals of Ontology Alignment is allowing semantic information systems to answer queries independently of the ontology from which the query terms have been taken. The common way to proceed in most part of existing systems is to materialize the obtained mappings into new statements (e.g. *owl:equivalentClass*), and then let the inference engine do its task. However, not all the mappings have the same level of confidence, and deciding which to include and which not becomes a problem that is usually solved heuristically.

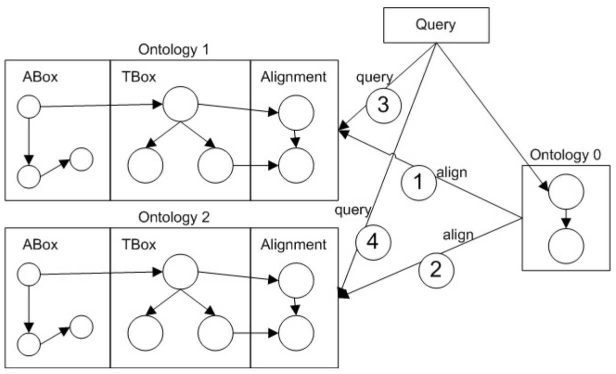


Figure 7.1: Workflow of the query answering process

7.2.1 Uncertain mappings

In traditional semantic information systems, if a statement does not satisfy the exact constraints of a query, then it is not included in the result. However, there are some situations when there are some degree of uncertainty over an statement. One of this situations could be an ontology alignment process, which returns a set of mappings with their respective level of confidence. Traditional description languages (e.g. OWL) or query languages (e.g. SPARQL), do not provide mechanisms to face this problem. However, some initiatives like Fuzzy OWL [141] or PR-OWL [27] are now working to include certainty in the Semantic Web.

[1]Abox and Tbox are used to describe two different types of statements in ontologies. Tbox statements describe a controlled vocabulary (e.g. a set of classes and properties) while Abox are statements about that vocabulary (instances).

PR-OWL (probabilistic OWL)

PR-OWL [27] is a novel ontology description language that extends OWL providing constructs for modelling uncertainty in ontologies. It allows moving beyond the current limitations of deterministic classical logic to a full first-order probabilistic logic. PR-OWL it is not limited to extending the attribute-value model by including syntax to describe probabilities, it goes beyond by allowing to represent complex Bayesian probabilistic models. PR-OWL has Multi-Entity Bayesian Networks (MEBN) as its underlying logical basis. This kind of networks combine Bayesian probability theory with classical First Order Logic.

f-OWL (fuzzy OWL)

Fuzzy OWL or simply f-OWL [141] is a fuzzy extension of OWL DL by adding degrees to OWL facts. Despite that the only syntactic change required is the addition of a membership degree, that ranges from 0 to 1, the semantics of f-OWL must be redefined. [141] describes the new semantics and also f-$\mathcal{SHOIN}$ as an extension of the $\mathcal{SHOIN}$ DL.

7.3 XML-RDF semantic integration

Interoperability among autonomous XML schemas has been one of the recent challenges of data integration. The success of the Resource Description Framework (RDF) [124] and its related technologies (RDFS [126], OWL [112]) has refuelled the problem by, on one hand, raising the necessity to establish interoperability mechanisms between XML schemas and RDF schemas (RDFS or OWL), and, on the other hand, opening the possibility to use RDFS/OWL ontologies as a solution for the semantic mapping problem among XML schemas.

One of the contributions of this thesis is strongly related by the research trend that tries to exploit the advantages of an XML-to-RDF mapping [54][65][5][84][87][117][132]. The XML-to-RDF mapping has been faced traditionally from what is known as the *structure-mapping* [97], that defines a direct way to map XML schema entities to RDF classes. Our contribution consists on exploring a different approach, the mapping of the general XML model to RDF. A more deep analysis of the related *state of the art* can be found in the *related work* section of Chapter 9.

7.4 Querying highly volatile and restricted Web data sources

The classical problems of data integration have well-known solutions like the GAV and LAV approaches or the MiniCon [119] and bucket [56] algorithms. However, the evolution of web-related technologies like XML and RDF, and the proliferation of new data sources and wrapper technologies suggest the reformulation of the old problems and solutions. One of the contributions of this thesis is related to using XQuery [154] in a LAV-based approach to query a set of spanish online newspapers. Recently Thomas Kabisch and Mattis Neiling [81] have used a very similar strategy but using RDF and RDQL to query data related to research papers from the best-known web sources. I borrow from them the name of

Query Tunneling to define this trend, which in fact is a simplification of the LAV approach but more appropriate for the very restricted web data interfaces.

7.5 Data integration in P2P

Peer-to-peer (P2P) architectures are becoming popular. They update the client-server model by eliminating the necessity of having central servers, reducing communication and storage costs and improving reliability. Some works have suggested that having a central schema for data is not a good idea in a P2P architecture. We can find an example of this in the *Peer-data management systems* (PDMS) [4][53][18]. In a PDMS participants do not need to agree on a central data schema, they can define their local semantic mappings to the most convenient peer, and queries can be answered by chaining mappings. This approach improves flexibility, allowing each peer to query the system using its own schema.

In [52] Alon Y. Halevy et al. described *Piazza: Data Management Infrastructure for Semantic Web*. Piazza is a PDMS based on the use of XML and XQuery but allows also the integration of RDF. In [4] are described some of the problems related to PDMS, focusing on data placement (also related to the Piazza system). They demonstrate that an intelligent materialization of views (replication) in some nodes in the network allows to improve performance and availability.

In [18] Philip A. Bernstein et al. described local relational models as a formalism for mediating between different peers in a PDMS, and an algorithm for answering queries using the formalism. In [103] is described the Edutella system, focused in the XML-RDF interoperability. It aims to provide query and storage services for RDF, but with the ability to use heterogeneous underlying sources. The RDF queries are reformulated to the underlying storage formats and query languages using canonical mappings (Edutella does not employ point-to-point mappings between nodes).

The Chatty Web [1] describes protocols for exchanging semantic mapping information in a decentralized fashion. Schemas and mappings are dynamically spread through the network by a *gossip* mechanism, and queries are routed and mapped using this information. Hyperion [82] faces the problem of mapping objects from different sources. It focuses on the use of relational tables and provides a theoretical model.

Finally, PeerDB [104] avoids schema mappings taking a different approach based on Information Retrieval algorithms for query reformulation. Each peer and each one of its attributes is associated with a set of keywords. Given a query over a peer schema, PeerDB reformulates the query into other peer schemas by matching the keywords associated with the attributes of the two schemas.

Chapter 8

Problem Statement

This thesis faces two specific problems within the general and old research topic of Data Integration. In general, integration of multiple data sources aims at giving a unified view over a set of pre-existent data. This entails to allow users a uniform access over the data without having to deal with the particularities of each source. Achieving this ambitious goal implies solving several problems that have defined different research topics.

8.1 Problem addressed 1: Semantic integration

Semantic heterogeneity is one of the key challenges in integrating and sharing data across disparate sources. Semantics refer to meaning, in contrast to syntax that refers to structure. In the database area, semantics can be regarded as people's interpretation of data and schema items according to their understanding of the world in a certain context. Semantic integration is the research area focused in reconciling data from autonomous sources using ontologies or other semantic-based tools.

This thesis aims to contribute to this research trend by providing solutions to two problems:

1. XML-RDF Semantic Integration: How to take profit from ontologies to integrate XML data from disparate schemas? How to query XML data related to multiple schemas but also to one or more ontologies? It is possible to do this and keep using conventional XML query languages like XPath or XQuery?

2. OWL Ontology Alignment: Can a rigorous and scalable semantic similarity measure be defined for OWL ontologies? Can an ontology alignment process successfully work directly over the RDF labeled directed graph, or it is better to process the equivalent bipartite graph?

8.2 Problem addressed 2: Heterogeneous query interfaces

Within the old data integration LAV (Local-As-View) approach, some solutions were provided to the problem of how an initial query, targeting a logical mediated schema, must be translated into queries over a set of different autonomous data sources. The old solutions, generally complex, were based on Datalog, and focus on answering expressive queries over heterogeneous but rich query interfaces. The evolution of the Web and its related technologies, like XML, allows to reformulate this old problem, that is the basis of the second main contribution of this work:

1. Can XML-technologies and a strategy based on the reprocessing of results be a practical solution for web-based data integration systems? How this approach can be instantiated to develop specific applications?

Part II

Heterogeneous Data Models and Schemas: Semantic Integration

Chapter 9

XML Semantic Integration: A Model Mapping Approach

This work describes 1) Why an ontology-aware XPath processor (or an XQuery engine that makes use of it) can be a natural and powerful tool for processing metadata, 2) How a processor with such behaviour has been implemented using Description Logics (materialised in RDFS and OWL constructs) and 3) A real application scenario in the Digital Rights Management (DRM) domain. We present the architecture of a schema-aware and ontology-aware XPath processor that acts over an RDF mapping of XML. Translating XML documents to RDF permits taking profit from the powerful tools of Description Logics allowing XML documents interoperate at the semantic level. We test our approach in the Digital Rights Management (DRM) domain, where some organizations are involved in standardization or adoption of rights expression languages (REL). We explore how a schema-aware and ontology-aware XPath/XQuery processor can be used in two of the main REL initiatives (MPEG-21 REL and ODRL).

9.1 Already published work

Large portions of this chapter have appeared in the following papers:

Tous R., García R., Rodríguez E., Delgado J. "Architecture of a Semantic XPath Processor. Application to Digital Rights Management", 6th International Conference on Electronic Commerce and Web Technologies EC-Web 2005. August 2005 Copenhagen, Denmark. Lecture Notes in Computer Science, Vol. 3590 (2005), pp. 1-10. ISSN: 0302-9743

Tous R., Delgado J. "A Semantic XPath processor". InterDB 2005 International Workshop on Database Interoperability. ELSEVIER's Electronic Notes in Theoretical Computer Science 2005

Tous R., Delgado J. "RDF Databases for Querying XML. A Model-mapping Approach". DISWeb 2005 International Workshop Data Integration and the Semantic Web. Procedings of the CAiSE'05 Workshops. Faculdade de Engenharia da Universidade do Porto. ISBN 972-752-077-4 Pages: 363 - 377

Tous R., Delgado J. “Using OWL for Querying an XML/RDF Syntax”. WWW’05: Special interest tracks and posters of the 14th international conference on World Wide Web. Chiba, Japan. Pages: 1018 - 1019. ACM Press 2005. ISBN:1-59593-051-5. http://doi.acm.org/10.1145/1062745.1062847

9.2 Introduction

The most part of XML-based applications make use of one or more XML schemas for instance validity check. In addition to defining a valid structure for the documents, generally the schemas define also inheritance hierarchies among types and element names (to rationalize the writing of the schemas and the instances respectively). However, sometimes it is necessary to consider this information not only for validation but also when evaluating queries over the XML data. Today it is also becoming common the use of RDFS/OWL ontologies to define semantic connections among application concepts. In some cases, the ontologies define relationships that are relevant for query evaluation (equivalences among names, subclassing, transitiveness, etc.). Unfortunately, all this structural and semantic knowledge is hard to access for developers, because it requires a specific treatment, like defining multiple extra queries for the schemas or using complex RDF tools to access the ontologies information.

To overcome this situation we present the architecture of a schema-aware and ontology-aware XPath/XQuery processor. The processor can be fed with an unlimited set of XML schemas and RDFS/OWL ontologies and will resolve the queries taking in consideration the structural and semantic connections described in them. To achieve this goal, the processor acts over an RDF mapping of XML, contributing to a recent research trend that defines an XML-to-RDF mapping allowing XML documents interoperate at the semantic level. We use a *model-mapping approach* to represent instances of XML and XML Schema in RDF. This representation retains the node order, in contrast with the usual *structure-mapping* approach, so it allows a complete mapping of all XPath axis.

This chapter is structured in three main blocks. First, we describe some related work to help identifying the problem and the relevance of the contribution. Second, we describe the architecture of the semantic XPath processor and some implementation details. Third, we apply our approach to a plausible usage scenario, the Digital Rights Management (DRM) domain. We explore how an XPath/XQuery processor with semantic behaviour can be used for processing licenses from two of the main Rights Expression Language (REL) initiatives (MPEG-21 REL and ODRL).

9.3 Related work

9.3.1 The query rewriting approach

There is a previous work that also pursues the target to achieve a *semantic* behaviour for XPath/XQuery. This approach is described in [91], and also shares with ours the translation from XML Schemas to OWL. Because the authors do not attempt to provide a new XQuery implementation, they use the obtained ontology as a guidance to rewrite the original *semantic* XQuery instance (they call it Semantic Web Query Language or SWQL) to a conventional XQuery instance.

The difference between this approach and ours is that our work does not describe a new query language and does not need a translation between the semantic queries and XPath/XQuery expressions. We have developed a new XPath processor that directly manipulates conventional XPath instances but taking in consideration the semantic relationships defined in the schemas and/or ontologies and related to the names involved in the query. The processor can be embedded into a conventional XQuery processor to obtain the *semantic* behaviour also for XQuery.

9.3.2 Other related work. Model-mapping vs. Structure-mapping

The key element of our work is the mapping of the XML and XML Schema models to OWL. The origins of this approach can be found in a research trend that tries to exploit the advantages of an XML-to-RDF mapping [54][65][5][84][87][117][132]. However, the concepts of *structure-mapping* and *model-mapping* are older. In 2001, [97] defined these terms to differentiate between works that map the structure of some XML schema to a set of relational tables (element names become table names) and works that map the XML model to a general relational schema (a small number of tables representing elements, atributes and relationships, element names become just field values) respectively.

More recently, [84] takes a *structure-mapping* approach and defines a direct way to map XML documents to RDF triples ([65] classifies this approach as *Direct Translation*). [54], [65], and [5] take also a *structure-mapping* approach but focusing on defining semantic mappings between different XML schemas ([65] classifies their own approach as *High-level Mediator*). They also describe some simple mapping mechanisms to cover just a subset of XPath constructs. Other authors like [87] or [117] take a slightly different strategy (though within the *structure-mapping* trend) and focus on integrating XML and RDF to incorporate to XML the inferencing rules of RDF (strategies classified by [65] as *Encoding Semantics*). Finally it's worth mention the RPath initiative [132], that tries to define an analogous language to XPath but for natural (not derived from XML) RDF data (this last work doesn't pursue interoperability between models or schemas).

9.4 Architecture of the semantic XPath processor

9.4.1 Overview

Figure 9.1 outlines how the schema-aware and ontology-aware XPath processor works. The key issue is the XML-to-RDF mapping, already present in other works, but that we face from the *model-mapping* approach. In contrast with the *structure-mapping* approach, that maps the specific structure of some XML schema to RDF constructs, we map the XML Infoset [68] using RDFS and OWL axioms based on the already existing W3C's RDFS informative representation [147]. This allows us to represent any XML document without any restriction and without losing information about node-order. We use the same approach with XSD, obtaining an RDF representation of the schemas, as we will explain later. Incorporating alternative OWL or RDFS ontologies is straightforward, because they

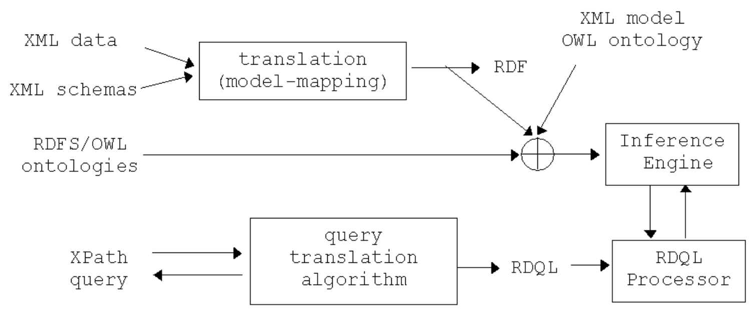


Figure 9.1: Semantic XPath processor architecture overview

are already compatible with the inference engine. In the figure we can see also that an OWL representation of the XML model is necessary. This ontology allows the inference engine to correctly process the different XPath axis and understand how the XML elements relate to the different XSD constructs.

Example 9.4.1. Let us see a simple example. Take the following XML document describing two movies:

```
 <movies>
  <movie id="m1">
    <title>Blade Runner</title>
    <year>1982</year>
    <director id="d1">
     <name>Ridley Scott</name>
    </director>
  </movie>
  <movie id="m2">
    <title>Paris, Texas </title>
    <year>1984</year>
    <director id="d2">
     <name>Wim Wenders</name>
    </director>
  </movie>
</movies>
```

And also its attached XML schema describing the valid structure for all "movies" documents:

```
<xs:schema>
  <xs:element name="movies">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="movie">
          <xs:complexType>
```

```
            <xs:sequence>
              <xs:element name="title"/>
              <xs:element name="year"/>
              <xs:element name="director">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="name"/>
                  </xs:sequence>
                  <xs:attribute name="id"/>
                </xs:complexType>
              </xs:element>
            <xs:attribute name="id"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

The object-oriented nature of some XML Schema constructs allows using them to increase the interoperability of applications or to fix interoperability problems in an elegant way. For example, the *substitutionGroup* inheritance mechanism can be used to bind the names of two different XML languages. The previous schema defines the elements *movies, movie, title, year*, etc. It could be interesting in some context to have the possibility to write the element and attribute names in a language different from English. We can generate a schema that binds the different names from the Spanish version to the (master) English version:

```
<xs:schema>
  <xs:element name="películas" substitutionGroup='movies'>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="película" substitutionGroup='movies'>
          <xs:complexType>
            <xs:sequence>
              <xs:element name="título" substitutionGroup='title'/>
              <xs:element name="año" substitutionGroup='year'/>
              <xs:element name="director" substitutionGroup='director'>
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="nombre" substitutionGroup='name'/>
                  </xs:sequence>
                  <xs:attribute name="id"/>
                </xs:complexType>
              </xs:element>
          <xs:attribute name="id"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

Now, using our schema-aware XPath processor, if we ask for */movie/country* we will obtain the same as for the */pelicula/pais* (independently if the XML instance is written in English or in Spanish). So, we can develop applications that are not tied to a particular schema but to an *global* one. This feature allows using XML schemas (or also OWL ontologies) to define semantic relationships between other XML schemas or ontologies, and to issue XPath queries that will be solved accordingly. This is just one of the features of the approach in a trivial scenario, but serves to illustrate the idea.

9.4.2 OWL. An ontology web language

The OWL Web Ontology Language, being produced by the W3C Web Ontology Working Group (WebOnt), is a language for defining and instantiating Web ontologies. The language can be used to formalize a domain by defining classes, properties of those classes and individuals. With the information of a domain in a machine-understandable format, inference engines or other application can reason about the different classes and individuals, deriving logical consequences, i.e. facts not literally present in the ontology, but entailed by the semantics.[1]

9.4.3 An OWL ontology for the XML model (XML/RDF Syntax)

Instead of taking the intuitive *structure-mapping* approach to transform a XML document in a set of RDF triplets like [65], we tried to represent the XML Infoset [68] using an OWL ontology based on the already existing W3C's [147]. This allows us to represent any XML document without any restriction and without losing information about node-order. Fig. 9.3 shows graphically how the example of fig. 9.2 will be represented using the classes and properties defined with OWL. The descendants of the class *node* (*document*, *element*, *attribute* and *textNode*) in conjunction with the ObjectProperty *childOf* are the main building blocks of the document tree, while the ObjectProperty *preceding-sibling* is necessary to preserve the node order.

Following we include the OWL ontology to show the details. We take profit from the expressive power of OWL to define properties like *parentOf*, *descendant*, *ancestor*, *descendantOrSelf*, *ancestorOrSelf*, *immediateFollowingSibling*, *followingSibling*, *following*, *precedingSibling*, and *preceding* just in terms of the two primitives *childOf* and *immediatePrecedingSibling*. This will be of great help later when we translate an XPath query to a RDQL query for the RDF-representation of the XML data. For e.g., we define *descendant* as a superset of *childOf*, which itself is defined as the inverse of *parentOf*. All these properties do

[1]see the *Background Information* chapters for a brief introduction to OWL

not need to be present in the representation because they will be deduced by the inference engine when processing the queries. A simplified description of the ontology in Description Logics syntax ($\mathcal{SHIQ}$-like style [62]) would be:

$$\begin{aligned}
Document &\sqsubseteq Node\\
Element &\sqsubseteq Node\\
TextNode &\sqsubseteq Node\\
childOf &\sqsubseteq descendant\\
parentOf &\sqsubseteq ancestor\\
childOf &\equiv parentOf^{-}\\
&\mathcal{T}rans(ancestor)\\
ancestor &\sqsubseteq ancestorOrSelf\\
self &\sqsubseteq descendantOrSelf\\
self &\sqsubseteq ancestorOrSelf\\
self &\equiv sameAs\\
immediatePrecedingSibling &\sqsubseteq precedingSiblinng\\
immediateFollowingSibling &\sqsubseteq followingSibling\\
immediatePrecedingSibling &\equiv immediateFollowingSibling^{-}\\
&\mathcal{T}rans(followingSibling)
\end{aligned}$$

Fig. 9.3 shows graphically how the example of fig. 9.2 will be represented using the classes and properties defined with OWL.

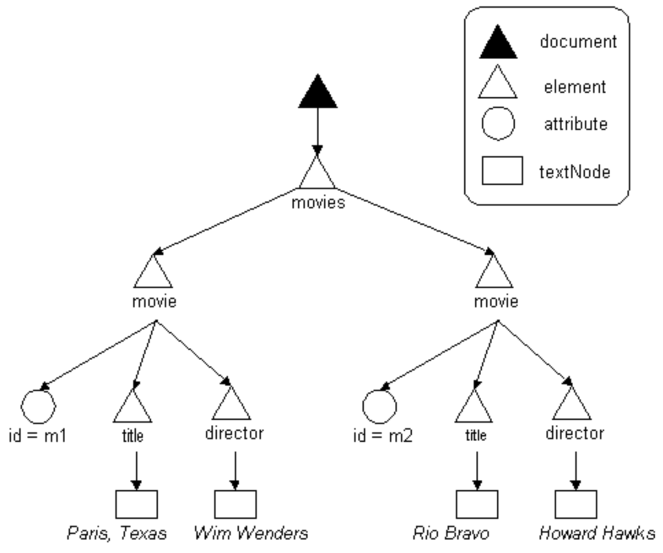


Figure 9.2: XML simple example describing two movies

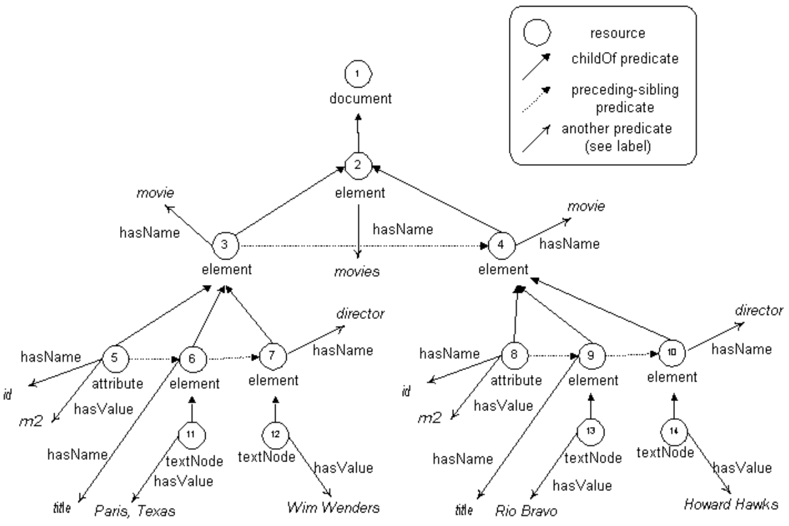


Figure 9.3: RDF graph for movies example

9.4.4 XPath

While the official definition of XPath [69] remains saying *"XPath is a language for addressing parts of an XML document"*, that was strictly correct for XPath 1.0, the version 2.0 cannot be defined just that way. Because the syntax of XPath 2.0 is a compact-version of XQuery 2.0, it better would be defined as e.g. *"a sequence-based language for querying and processing XML"*. XPath uses a compact, non-XML syntax to facilitate use of XPath within URIs and XML attribute values. XPath operates on the abstract, logical structure of an XML document, rather than its surface syntax.

[69] says that XPath 2.0 has been designed to be embedded in another host language such as XSLT 2.0 [155] or XQuery 1.0 [154]. However the relation of XPath and these two languages differs. XPath 2.0 and XQuery 1.0 have the same semantics, defined by XQuery 1.0 and XPath 2.0 Formal Semantics [153]. [69] says that *"XQuery 1.0 is an extension of XPath 2.0"*. So one can talk about the same language with two syntaxes, one with the SQL flavour (XQuery), and the compact version (XPath) to be embedded in a host language (XSLT).

9.4.5 XPath data model

[152] specifies the XQuery 1.0 and XPath 2.0 data model. It defines the information contained in the input to the host language in which XPath is embedded and also all permissible values of XPath expressions. The data model is based on the XML Infoset [68]. The following definitions (extracted from [152] and not comprehensive) describe the key elements of the XPath data model:

1. Every instance of the data model is a sequence.

2. A sequence is an ordered collection of zero or more items.

3. A sequence cannot be a member of a sequence.
4. An item is either a node or an atomic value
5. Every node is one of the seven kinds of nodes (document, element, attribute, text, namespace, processing instruction, and comment).

So, the basic building block of the data model is the *sequence*. This is an important difference with respect to Xpath 1.0, in which the basic constructs were *node-sets* (without duplicates). Sequences can contain duplicates but not other sequences, combining sequences always produce a flattened sequence instead of a nesting.

9.4.6 XPath syntax

The grammar rules of XPath 2.0 have increased in complexity, since now supports *for* expressions, conditionals, intersections, unions and differences among other constructs. Here we are going to describe just the rules that are shared with the version 1.0, focusing in location paths (now *PathExpr*). The partial Backus-Naur Form (BNF) rules for an XPath's *expression* are:

```
Expr ::= ExprSingle ("," ExprSingle)*
ExprSingle ::= ForExpr | QuantifiedExpr | IfExpr | OrExpr
OrExpr ::= AndExpr ( "or" AndExpr )*
AndExpr ::= PathExpr ( "and" PathExpr )*
```

The basic building block of the syntax is the *expression*, which is a string of Unicode characters. For this work we are going to consider just expressions consisting on a single *PathExpr*, the basic construct to address parts of an XML document. The BNF rules for a *PathExpr* are:

```
PathExpr ::= RelativePathExpr | ("/" RelativePathExpr)?
RelativePathExpr ::=  AxisStep "/" (AxisStep)*
AxisStep ::= (ForwardStep | ReverseStep) Predicate*
ForwardStep ::= (ForwardAxis NodeTest) | AbbrevForwardStep
AbbrevForwardStep ::= "@"? NodeTest
ReverseStep ::= (ReverseAxis NodeTest) | AbbrevReverseStep
AbbrevReverseStep ::= ".."
NodeTest ::= KindTest | NameTest
NameTest ::= QName | "*"
KindTest ::= "node()" | "text()" | ...
Predicate ::= "[" Expr "]"
```

These rules, extracted from [69], were more simple and clear in the version 1.0. They say simply that a *PathExpr* is a sequence of steps (*axisStep*), each one composed of an axis (*ForwardAxis or ReverseAxis*), a *NodeTest* and a list of *Predicates*. Axis are the key element, because define the direction of each step. There are 13 different axis:

```
ForwardAxis ::= <"child" "::">
              | <"descendant" "::">
              | <"attribute" "::">
              | <"self" "::">
              | <"descendant-or-self" "::">
              | <"following-sibling" "::">
              | <"following" "::">
              | <"namespace" "::">
ReverseAxis ::= <"parent" "::">
              | <"ancestor" "::">
              | <"preceding-sibling" "::">
              | <"preceding" "::">
              | <"ancestor-or-self" "::">
```

Table 9.1 gives the meaning of each axis. Some example XPath queries for an XML document describing movies could be:

```
/child::movies/child::movie/child::title
(in abbreviated form /movies/movie/title)
/descendant-or-self::title
(in abbreviated form //title)
/child::movies/child::movie[@id='m1']
                 /following-sibling::node()
(in abbreviated form /movies/movie[@id='m1']
                 /following-sibling)
```

9.4.7 XPath Formal semantics

XPath can be formally defined by describing the operations on this data model. It is not a coincidence that some of the axioms are already present in the XML/RDF ontology, because they map directly to XML primitives (e.g. *child*). First we must define the function E, corresponding to the *XPathExpr* rule from the EBNF grammar [69].

$$E : Path \rightarrow Node \rightarrow sequence(Node)$$

$$\begin{aligned} E[[e_1/e_2]]_x &= \{x_2 \mid x_1 \in E[[e1]]_x \wedge x_2 \in E[[e2]]_{x1}\} \\ E[[a :: t]]_x &= \{x_1 \mid x_1 \in A_a(x) \wedge T_t(x_1)\} \\ E[[e[p]]]_x &= \{x_1 \mid x_1 \in E[[e]]_x \wedge P[[p]]_{x_1}\} \end{aligned}$$

The function A_a describes both the *ForwardAxis* and the *ReverseAxis* rules from the grammar.

$$A_a :\rightarrow Node \rightarrow sequence(Node)$$

Table 9.1: XPath axis

child	All children of the context element (attributes cannot have children)
descendant	The descendants of the context node (the children, the children of the children, and so on)
parent	The parent of the context node.
ancestor	The ancestors of the context node (the parent, the parent of the parent, and so on)
following-sibling	Those children of the context node's parent that occur after the context node in document order
preceding-sibling	Those children of the context node's parent that occur before the context node in document order
following	All nodes that are descendants of the root of the tree in which the context node is found, are not descendants of the context node, and occur after the context node in document order
preceding	All nodes that are descendants of the root of the tree in which the context node is found, are not ancestors of the context node, and occur before the context node in document order
attribute	The attributes of the context node
namespace	Namespace nodes
self	The context node
descendant-or-self	The context node and the descendants of the context node
ancestor-or-self	The context node and the ancestors of the context node

$$\begin{aligned}
A_{child}(x) &= \{x_1 \mid childOf(x_1, x)\} \\
A_{descendant}(x) &= \{x_1 \mid childOf(x_1, x) \vee \\
&\quad (childOf(x_2, x) \\
&\quad \wedge x_1 \in A_{descendant}(x_2))\} \\
A_{descendant-or-self}(x) &= \{x\} \cup \{x_1 \mid x_1 \in A_{descendant}(x)\} \\
A_{parent}(x) &= \{x_1 \mid childOf(x, x_1)\} \\
A_{ancestor}(x) &= \{x_1 \mid childOf(x, x_1) \vee \\
&\quad (childOf(x, x_2) \wedge x_1 \in A_{ancestor}(x_2))\}
\end{aligned}$$

$$\begin{aligned}
A_{ancestor-or-self}(x) &= \{x\} \cup \{x_1 \mid x_1 \in A_{ancestor}(x)\} \\
A_{preceding-sibling}(x) &= \{x_1 \mid precedingSibling(x_1, x)\} \\
A_{preceding}(x) &= \{x_1 \mid x_1 \in A_{descendant-or-self}(x_2) \\
&\quad \wedge x_2 \in A_{preceding-sibling}(x_3)\} \\
&\quad \wedge x_3 \in A_{ancestor-or-self}(x)\} \\
A_{following-sibling}(x) &= \{x_1 \mid precedingSibling(x, x_1)\} \\
A_{following}(x) &= \{x_1 \mid x_1 \in A_{descendant-or-self}(x_2) \\
&\quad \wedge x_2 \in A_{following-sibling}(x_3)\} \\
&\quad \wedge x_3 \in A_{ancestor-or-self}(x)\} \\
A_{attribute}(x) &= \{x_1 \mid attributeOf(x_1, x)\} \\
A_{attribute}(x) &= \{x_1 \mid namespaceOf(x_1, x)\}
\end{aligned}$$

The function T describes the *NodeTest* rule from the grammar.

$$T : NodeTest \to Node \to Boolean$$

$$\begin{aligned}
T_*(x) &= \{true\} \\
T_n(x) &= \{hasName(x, n)\} \\
T_{node()}(x) &= \{type(x,' node')\} \\
T_{text()}(x) &= \{type(x,' textNode')\} \\
T_{element()}(x) &= \{type(x,' elementNode')\}
\end{aligned}$$

The function P describes the *Predicates* rule from the grammar. There are a lot of different predicates but defining all is out of the scope of this document. As an example we define here the predicate that expresses the existence of a specific sub-tree as a condition.

$$P : Predicate \to Node \to Boolean$$

$$P[[p]]_x = \{\exists x_1 \in E[[p]]_x\}$$

9.4.8 RDQL A Query Language for RDF

RDQL [127] is the popular RDF query language from HP Labs Bristol. RDQL is an implementation of the SquishQL [139] RDF query language, which itself is derived from rdfDB [125]. The specification of RDQL was submitted to the W3C in 9 January 2004, and has an enormous influence to the new W3C's RDF query language, SPARQL [138]. However we have chosen RDQL instead of SPARQL because of the existence of a mature query processor as the Jena API [76]. The results obtained are extensible (and we plan to do this explicit when tools are available) to the new W3C's language.

An RDF model is a graph, often expressed as a set of triples. An RDQL query consists of a graph pattern, expressed as a list of triple patterns. Each triple pattern is comprised of named variables and RDF values (URIs and literals). An RDQL query can additionally have a set of constraints on the values of those variables, and a list of the variables required in the answer set. An example RDQL query could be:

```
SELECT ?book
WHERE  (?book, <somelibrary:year>, ?year)
AND    ?year >= 2004
USING  somelibrary FOR <http://example.somelibrary.org/books#>
```

This sample query will return all the RDF triples with a predicate somelibrary:year and a literal object consisting on an integer equal or greater than 2004. A complete explanation of the language can be found in [127].

9.4.9 XPath translation to RDQL

Some works face the problem to execute XPath queries over RDF data. Most of them (like [65]) take a *structure-mapping* approach and describe some simple mapping mechanisms to cover just a subset of XPath constructs (as mentioned before it is not feasible to map the constructs based on node-order in a structure-mapping approach). Another works, like the RPath initiative [132], try to define an analogous language to XPath but for natural (not derived from XML) RDF data.

Our strategy is radically different because we transform a XPath query into a RDQL query that we execute over an exact (and not just an intuitive mapping) RDF representation of the input XML data. This makes feasible the mapping of all XPath constructs in a natural and elegant way.

Each XPath *axis* can be mapped into one or more triple patterns of the target RDQL [127] query. Analogously each *nodetest* and *predicate* can be mapped also with just one ore more triple patterns. The output RDQL query always takes the form:

```
SELECT *
WHERE
  (?v1, <rdf:type>, <xmloverrdf:document>)
  [triple pattern 2]
  [triple pattern 3]
  ...
  [triple pattern N]
USING
  xmloverrdf  FOR <http://dmag.upf.edu/xml#>
```

The translation can be deduced from the XPath formal semantics. For example, the *following* axis is described as:

$$\begin{aligned} A_{following}(x) = \{x_1 \mid x_1 \in\ & A_{descendant-or-self}(x_2) \\ & \wedge x_2 \in A_{following-sibling}(x_3)\} \\ & \wedge x_3 \in A_{ancestor-or-self}(x)\} \end{aligned}$$

So the *following* axis must be translated to:

```
(?vi, <xmloverrdf:ancestor-or-self>, ?vi-1)
 i = i + 1
(?vi, <xmloverrdf:following-sibling>, ?vi-1)
i = i + 1
(?vi, <xmloverrdf:descendant-or-self>, ?vi-1)
i = i + 1
```

There are also simple conversion rules for all *nodeTests* and *predicates* but we omit them to save space. The notation used includes variable names like *vi* and *vi-1* where *i* begins with value *2* (because of the first triple pattern is always the same as shown before). So if we would have just the expression:

```
/child::movies/child::movie
```

We will translate the first child axis to:

```
(?v2, <xmloverrdf:childOf>, ?v1)
```

The first node test to:

```
(?v2, <xmloverrdf:hasName>, <http://dmag.upf.edu/xmlrdf/names#movies>)
```

The second child axis to:

```
(?result, <xmloverrdf:childOf>, ?v2)
```

And the second node test to:

```
(?result, <xmloverrdf:hasName>, <http://dmag.upf.edu/xmlrdf/names#movie>)
```

The complete WHERE clause will appear as:

```
WHERE
  (?v1, <rdf:type>, <xmloverrdf:document>)
  ,(?v2, <xmloverrdf:childOf>, ?v1)
  ,(?v2, <xmloverrdf:hasName>, <http://dmag.upf.edu/xmlrdf/names#movies>)
  ,(?result, <xmloverrdf:childOf>, ?v2)
  ,(?result, <xmloverrdf:hasName>, <http://dmag.upf.edu/xmlrdf/names#movie>)
```

9.4.10 Example results

An example query could be:

```
/child::movies/child::movie/child::title
(in abbreviated form /movies/movie/title)
```

That is translated to:

```
SELECT *
WHERE
        (?v1, <rdf:type>, <xmloverrdf:document>)
        , (?v2, <xmloverrdf:childOf>, ?v1)
        , (?v2, <xmloverrdf:hasName>, "movies")
        , (?v3, <xmloverrdf:childOf>, ?v2)
        , (?v3, <xmloverrdf:hasName>, "movie")
        , (?result, <xmloverrdf:childOf>, ?v3)
        , (?result, <xmloverrdf:hasName>, "title")

Result: 6, 9 (node numbers, see figure)
```

9.5 Incorporating schema-awareness

9.5.1 Mapping XML Schema to RDF

In our ontology for the XML model, the object of the *hasName* property is not a literal but a resource (an RDF resource). This key aspect allows to apply to *hasName* all the potential of the OWL relationaships (e.g. defining ontologies whith names relationships). So, if we want our XPath processor to be schema-aware, we just need to translate the XML Schema language to RDF, and to add to our XML/RDF Syntax ontology the necessary OWL constructs that allow the inference engine to understand the semantics of the different XML Schema components. The added axioms in Desctiption Logics syntax ($\mathcal{SHIQ}$-like style [62]) would be:

$$\begin{aligned}
hasName &\sqsubseteq fromSubstitutionGroup \\
&\quad \mathcal{T}rans(fromSubstitutionGroup) \\
hasName &\sqsubseteq fromType \\
&\quad \mathcal{T}rans(fromType) \\
fromType &\sqsubseteq subTypeOf
\end{aligned}$$

9.5.2 A simple example of schema-aware XPath processing

The next example ilustrates the behaviour of our processor in a schema-related XPath query. Take this simple XML document:

```
<A>
  <B id='B1' />
  <B id='B2'>
   <C id='C1'>
    <D id='D1'></D>
   </C>
  </B>
  <B id='B3'/>
</A>
```

And its attached schema:

```
<schema>
 <complexType name='BType'>
   <complexContent>
      <extension base='SUPERBType'></extension>
   </complexContent>
 </complexType>
 <element name='B'
          type='BType' substitutionGroup='SUPERB' />
</schema>
```

When evaluating the XPath query *//SUPERB*, our processor will return the elements with IDs 'B1', 'B2' and 'B3'. These elements have a name with value 'B', and the schema specifies that this name belongs to the substitution group 'SUPERB', so they match the query. Also, when evaluating the query *//SUPERBType*, the processor will return 'B1', 'B2' and 'B3'. It assumes that the query is asking for elements from the type SUPERBType or one of its subtypes.

9.5.3 Complete XSD to OWL Mapping

The previous XML Schema (XSD) to RDF mapping is partial in the sense that it just maps the XML Schema semantics that are needed in order to make the XPath processor XSD semantics aware. There is also a more complete XML Schema to OWL mapping (XSD2OWL) that is responsible for capturing almost all the schema implicit semantics. This semantics are determined by the combination of XML Schema constructs. The XSD2OWL mapping is based on translating these constructs to the OWL ones that best capture their semantics. The informal semantics of XML Schema constructs are presented in Table 9.2 and then used to guide the XML Schema to OWL mappings shown in Table 9.3.

The XSD2OWL mapping is quite transparent and captures a great part XML Schema semantics. The same names used for XML constructs are used for OWL ones, although in the new namespace defined for the ontology. XSD and OWL constructs names are identical; this usually produces uppercase-named OWL properties because the corresponding element name is uppercase, although this is not the usual convention in OWL.

Therefore, XSD2OWL produces OWL ontologies that make explicit the semantics of the corresponding XML Schemas. The only caveats are the implicit order conveyed by xsd:sequence and the exclusivity of xsd:choice.

For the first problem, owl:intersectionOf does not retain its operands order, there is no clear solution that retains the great level of transparency that has been achieved. The use of RDF Lists might impose order but introduces ad-hoc constructs not present in the original metadata. Moreover, as it has been demonstrated in practise, the elements ordering does not contribute much from a semantic point of view. For the second problem, owl:unionOf is an inclusive union, the solution is to use the disjointness OWL construct, owl:disjointWith, between all union operands in order to make it exclusive.

The resulting OWL ontology is OWL-Full because the XSD2OWL translator has employed rdf:Property for properties to cope with the fact that there are properties that have

Table 9.2: XML Schema informal semantics

XML Schema	**Shared informal semantics**
element | attribute	Named relation between nodes or nodes and values
element@substitutionGroup	Relation can appear in place of a more general one
element@type	The relation range kind
complexType | group | attributeGroup	Relations and contextual restrictions package
complexType//element	Contextualised restriction of a relation
extension@base | restriction@base	Package concretises the base package
@maxOccurs @minOccurs	Restrict the number of occurrences of a relation
sequence choice	Combination of relations in a context

Table 9.3: XSD2OWL translations for the XML Schema constructs

XML Schema	**OWL**
element | attribute	rdf:Property owl:DatatypeProperty owl:ObjectProperty
element@substitutionGroup	rdfs:subPropertyOf
element@type	rdfs:range
complexType | group | attributeGroup	owl:Class
complexType//element	owl:Restriction
extension@base | restriction@base	rdfs:subClassOf
@maxOccurs @minOccurs	owl:maxCardinality owl:minCardinality
sequence choice	owl:intersectionOf 0wl:unionOf

both data type and object type ranges as specified in the XML Schema for the corresponding xsd:element.

The full mapping facilitates the implementation of XSD semantics-aware applications. These applications can use the XSD schema semantics formalised in the corresponding ontologies. This ontologies enable semantic XPaths but also they open the possibility to use other semantics-enabled tools to the XML domain, e.g. reasoning engines or ontology alignment solutions for schema integration.

For instance, this approach has already shown its usefulness in the Digital Rights Management (DRM) domain [48]. These ontologies have been then exploited for DRM Systems implementation [45] [46] and assisted negotiation of digital goods [47].

9.6 Implementation and performance

The work has been materialised in the form of a Java API. We have used the Jena 2 API [76] for RDQL computation and OWL reasoning. To process XPath expressions we have modified and recompiled the Jaxen XPath Processor [75]. An on-line demo can be found at *http://dmag.upf.edu/contorsion.*

9.6.1 Jena Inference Engine

The Jena API [76] provides a set of different inference engines or reasoners. We use the Jena's OWL reasoner to allow the RDQL query processor to derive additional RDF assertions from the base RDF data together with the XML/RDF ontology axioms. This reasoner includes rules for each one of the OWL/Lite constructs and also others, so it can be considered an incomplete implementation of OWL/Full. Table 9.4 enumerates the constructs supported by the OWL reasoner.

Table 9.4: OWL constructs supported by the Jena's OWL reasoner

Constructs
rdfs:subClassOf, rdfs:subPropertyOf, rdf:type
rdfs:domain, rdfs:range
owl:someValuesFrom, owl:allValuesFrom
owl:minCardinality, owl:maxCardinality, owl:cardinality
owl:intersectionOf
owl:equivalentClass, owl:disjointWith
owl:sameAs, owl:differentFrom, owl:distinctMembers
owl:Thing
owl:equivalentProperty, owl:inverseOf
owl:FunctionalProperty, owl:InverseFunctionalProperty
owl:SymmeticProperty, owl:TransitiveProperty
owl:hasValue

The OWL reasoner is built on top of a general purpose rule engine. This engine allows rule-based inference over RDF graphs, combining two different strategies. On one hand the engine uses *forward chaining* (specifically the *RETE* algorithm [40]) to precompute deductions. On the other hand it uses *backward chaining* (a Logic Programming strategy like Prolog) to answer the queries. The combination of these wwo strategies (*hybrid model*) is used by default, but the engine can be configured to run just one of them.

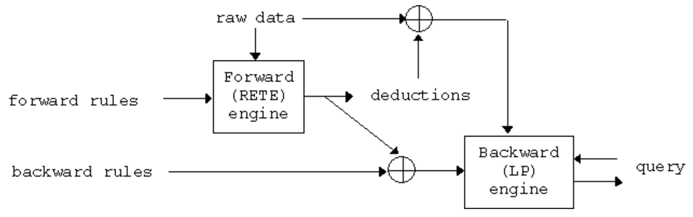


Figure 9.4: Jena hybrid execution model

The forward engine maintains a set of inferred statements in the deductions store. Forward rules can infer new data (deductions) and also other rules. When a query is formulated, the backward chaining LP engine applies the merge of the supplied and generated rules to the merge of the raw and deduced data.

The hybrid approach allows improving performance by reducing e.g. the generality of some backward rules that could be instantiated for a specific dataset. As an example, extracted from [77], consider the RDFS subPropertyOf entailments. A simple solution would involve the following backward rule:

```
(?a ?q ?b) <- (?p rdfs:subPropertyOf ?q), (?a ?p ?b) .
```

Of course the rule would work, but because the head is composed just by variables, every goal from the query will match. This will cause that the engine will have to test for subProperty relations for all possible goals. So, it makes sense to adapt this rule to a specific dataset before the backward process begin. We can try the following combination of a forward rule and a backward rule:

```
(?p rdfs:subPropertyOf ?q), notEqual(?p,?q)
                   -> [ (?a ?q ?b) <- (?a ?p ?b) ] .
```

The forward strategy would precompile all the declared subPropertyOf relationships into simple backward rules. These rules would only be fired if the goal references a property which actually has a sub property.

9.6.2 Performance

Though performance wasn't the target of the work, it is an important aspect of the processor. We have realised a performance test over a Java Virtual Machine v1.4.1 in a 2GHz Intel Pentium processor with 256Mb of memory. The final delay depends mainly on two variables, the size of the target documents, and the complexity of the query. Table 9.5 shows the delay of the inferencing stage for different document depth levels and also for some different queries.

The processor behaves well with medium-size documents and also with large ones when simple queries are used (queries that not involve transitive axis), but when document size grows the delay related to the complex queries increases exponentially. Some performance limitations of the Jena's OWL inference engine have been described in [78]. We are now working on this problem, trying to obtain a more scalable inference engine.

Table 9.5: Performance for different document depth levels

expression	5d	10d	15d	20d	20d (Xalan XPath processor)
/A/B	32ms	47ms	47ms	62ms	16ms
/A/B/following-sibling::B	125ms	46ms	48ms	47ms	15ms
/A/B/following::B	125ms	62ms	63ms	47ms	16ms
/A//B	172ms	203ms	250ms	219ms	31ms
//A//B	178ms	266ms	281ms	422ms	32ms

9.7 Testing in the DRM Application Domain

The amount of digital content delivery in the Internet has made Web-scale Digital Rights Management (DRM) a key issue. Traditionally, DRM Systems (DRMS) have dealt with this problem for bounded domains. However, when scaled to the Web, DRMSs are very difficult to develop and maintain. The solution is interoperability of DRMS, i.e. a common framework for understanding with a shared language and vocabulary. That is why it is not a coincidence that organisations like MPEG (Moving Picture Experts Group), OMA (Open Mobile Alliance), OASIS (Organization for the Advancement of Structured Information Standards), TV-Anytime Forum, OeBF (Open eBook Forum) or PRISM (Publishing Requirements for Industrial Standard Metadata) are all involved in standardisation or adoption of rights expression languages (REL). Two of the main REL initiatives are MPEG-21 REL [149] and ODRL [66].

Both are XML sublanguages defined by XML Schemas. The XML Schemas define the language syntax and a basic vocabulary. These RELs are then supplemented with what are called Rights Data Dictionaries [133]. They provide the complete vocabulary and a lightweight formalisation of the vocabulary terms semantics as XML Schemas or ad hoc ontologies. ODRL and MPEG-21 REL have just been defined and are available for their implementation in DRMS. They seem quite complete and generic enough to cope with such a complex domain. However, the problem is that they have such a rich structure that they are very difficult to implement. They are rich in the context of XML languages and the "traditional" XML tools like DOM or XPath. There are too many attributes, elements and complexTypes (see Table 9.6) to deal with.

9.7.1 Application to ODRL license processing

Consider looking for all constraints in a right expression, usually a rights license, that apply to how we can access the licensed content. This would require so many XPath

Table 9.6: Number of named XML Schema primitives in ODRL and MPEG-21 REL

	Schemas	**xsd:attribute**	**xsd:complexType**	**xsd:element**	**Total**
ODRL	EX-11	10	15	23	127
	DD-11	3	2	74	
MPEG-21	EL-R	9	56	78	330
	REL-SX	3	35	84	
	REL-MX	1	28	36	

queries as there are different ways to express constraints. For instance, ODRL defines 23 constraints: industry, interval, memory, network, printer, purpose, quality... This amounts to lots of source code, difficult to develop and maintain because it is very sensible to minor changes to the REL specs. Hopefully there is a workaround hidden in the language definitions.

As we have said, there is the language syntax but also some semantics. The *substitutionGroup* relations among *elements* and the *extension/restriction base* ones among *complexTypes* encode generalisation hierarchies that carry some lightweight, taxonomy-like, semantics. For instance, all constraints in ODRL are defined as XML elements substituting the *o-ex:constraintElement*, see Figure 9.5. The difficulty is that although this information

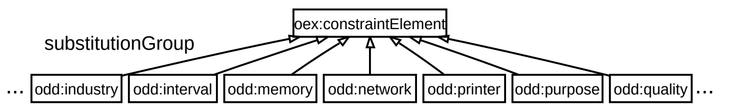


Figure 9.5: Some ODRL constraint elements defined as *substitutionGroup* of *constraintElement*

is provided by the XML Schemas, it remains hidden when working with instance documents of this XML Schemas. However, using the semantics-enabled XPath processor we can profit from all this information. As it has been shown, the XML Schemas are translated to OWL ontologies that make the generalisation hierarchies explicit, using *subClassOf* and *subPropertyOf* relations. The ontology can be used then to carry out the inferences that allow a semantic XPath like "//o-ex:constraintElement" to retrieve all *o-ex:constraintElement* plus all elements defined as its *substitutionGroup*.

9.7.2 Application to the MPEG-21 authorization model

MPEG-21 REL

In MPEG-21 standard the protection and governance of digital content are specified in MPEG-21 IPMP Components [70], MPEG-21 REL [128] and RDD [123] parts. MPEG-21 IPMP Components provides mechanisms to protect a digital item (DI) [29] and to associate licenses to the target of their governance, while MPEG-21 REL specifies the syntax and semantics of the language for issuing rights for users to act on DIs while MPEG-21 RDD comprises a set of terms to support the MPEG-21 REL. Suppose an MPEG-21 compliant peer that receives a protected and governed MPEG-21 DI, which consists of a digital asset and some metadata, as the information related to the tools to unprotect the asset and the conditions of use of this asset (see example). When processing this DI the first step is to obtain the IPMP metadata associated to the asset, then if any license that governs the protected asset is found, the application has to resolved if the user can exercise the requested action by means of the authorization mechanism defined in MPEG-21 REL, and if the result is positive the asset is unprotected and the action is exercised.

```
<DIDL>
  <Item>
    <Component>
      <Resource mimeType="application/ipmp">
        <ipmpdidl:ProtectedAsset mimeType="audio/mp3">
          <ipmpdidl:Identifier>
            <dii:Identifier> urn:mpegRA:mpeg21:dii:as002-11</dii:Identifier>
          </ipmpdidl:Identifier>
          <ipmpdidl:Info>
            <ipmpinfo:IPMPInfoDescriptor>
                <ipmpinfo:Tool> ... </ipmpinfo:Tool>
                 <ipmpinfo:RightsDescriptor>
                  <ipmpinfo:License>
                    <r:license> ... </r:license>
                  </ipmpinfo:License>
                </ipmpinfo:RightsDescriptor>
            </ipmp:IPMPInfoDescriptor>
          </ipmpdidl:Info>
          <ipmpdidl:Contents> EFJDV9FUV98JRF424U039RNCNK... </ipmpdidl:Contents>
        </ipmpdidl:ProtectedAsset>
      </Resource>
    </Component>
  </Item>
</DIDL>
```

In the scenario described above, the XPath processor is useful when implementing license based authorization mechanisms. MPEG-21 REL standard specification defines the authorization model, Figure 9.6, that makes use of the authorization request and story elements and resolves the question "Is a Principal authorized to exercise a Right such a Resource?". The XPath processor simplifies the implementation of the authorization algorithm because it allows to the application to quickly identify which elements are of a particular type in the licenses, authorization request and stories considered to resolve this algorithm. Therefore, when the application has to determine if a license or a grant within an authorization request or story has any element representing a resource, a principal or a condition, this process could result complex and costly if we don't use the XPath processor. A clear example is when we look for a resource in a license or grant element within an authorization request or story, if we don't have the capability to search for an element that its substitutionGroup is resource, then we have to look for one of the elements depicted in Figure 9.7. In the

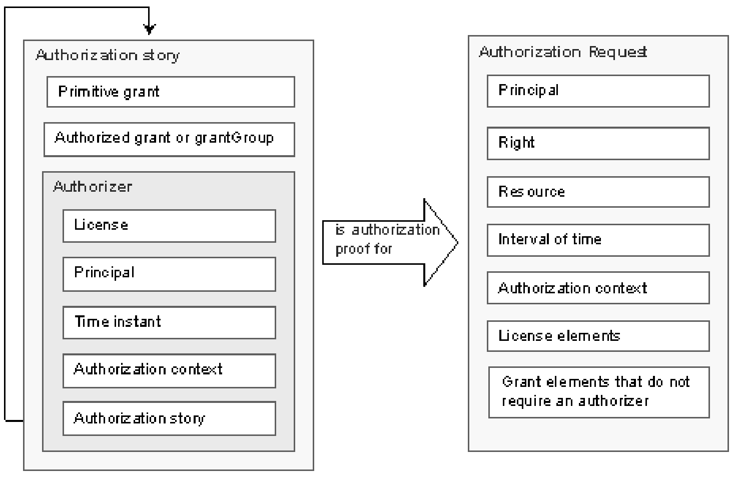


Figure 9.6: MPEG-21 REL authorization model

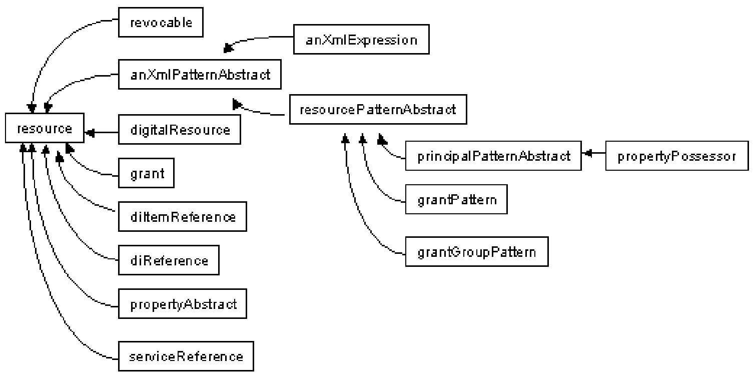


Figure 9.7: Example of resource elements

authorization process, the Xpath processor also is useful when comparing the right the user wants to exercise and the right in the user's license, because we have to take into account the rights lineage defined in RDD as described above.

MPEG-21 RDD

In order to interpret REL licenses, the semantic XPath processor help us when determining if the user has the appropriate rights taking into account the rights lineage defined in the RDD (Rights Data Dictionary).

In contrast with ODRL, that uses XMLSchemas both for the language and dictionary definitions, MPEG-21 has an ontology as dictionary (RDD). The semantics that it provides can also be integrated in our semantic XPath processor. To do that, the MPEG-21 RDD ontology is translated [74] to the ontology language used by the Semantic XPath Processor, i.e. OWL. Once this is done, this ontology is connected to the semantic formalisation build up from the MPEG-21 REL XML Schemas. Consequently, semantic XPath queries can also profit from the ad hoc ontology semantics. For instance, the acts taxonomy in MPEG-21 RDD, see Figure 9.8, can be seamlessly integrated in order to facilitate license checking implementation. Consider the scenario: we want to check if our set of licenses authorises us to uninstall a licensed program. If we use XPath, there must be a path to look for

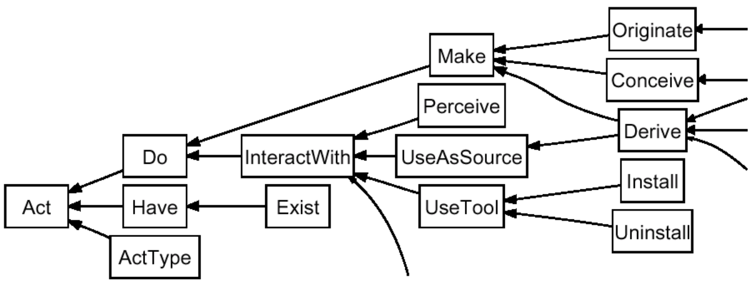


Figure 9.8: Portion of the acts taxonomy in MPEG-21 RDD

licenses that grant the *uninstall* act, e.g. "//r:license/r:grant/mx:uninstall". Moreover, as it is shown in the taxonomy, the *usetool* act is a generalisation of the *uninstall* act. Therefore, we must also check for licenses that grant us *usetool*, e.g "//r:license/r:grant/mx:uninstall". An successively, we should check for *interactwith*, *do* and *act*.

However, if we use a semantic XPath, the existence of a license that grants any of the acts that generalise *uninstall* implies that the license also states that the *uninstall* act is also granted. This is so because, by inference, the presence of the fact that relates the license to the granted act implies all the facts that relate the license to all the acts that specialise this act. Therefore, it would suffice to check the semantic XPath expression "//r:license/r:grant/mx:uninstall". If any of the more general acts is granted it would match. For instance, the XML tree */r:license/r:grant/dd:usetool* implies the trees */r:license/r:grant/dd:install* and */r:license/r:grant/dd:uninstall*.

9.8 Conclusions

In this chapter we have described a novel strategy for designing a schema-aware and ontology-aware XPath/XQuery processor. Such behaviour, that we have called *semantic*, can be used to transparently resolve queries over XML instances bound to schemas that define inheritance hierarchies among types and element names, or related to ontologies that define relationships that are relevant for the queries evaluation.

Our approach has consisted in mapping the XML and XSD models to OWL, allowing a complete translation from XML and XSD instances to RDF triples. The XPath expressions are translated to RDQL queries (we have provided a simple and elegant algorithm to do it) that are then resolved by an RDQL engine with OWL reasoning capabilities. The chosen representation retains the node order, in contrast with the usual *structure-mapping* approach, that maps the specific structure of some XML schema to RDF constructs. The work has been materialised in the form of a Java API. An on-line demo can be found at *http://dmag.upf.edu/contorsion.*

Finally, we have demonstrated how our approach can be useful in a plausible usage scenario, the Digital Rights Management domain, where the schema-aware and ontology-aware XPath/XQuery Processor has shown its benefits. The behaviour of the processor allows a transparent access to the semantics hidden in the schemas of the Rights Expression Languages, so we do not need to recode them. This allows developing software less coupled with the underlying specifications.

Chapter 10

A Vector Space Model for Semantic Similarity Calculation and OWL Ontology Alignment

Ontology alignment (or matching) is the operation that takes two ontologies and produces a set of semantic correspondences (usually semantic similarities) between some elements of one of them and some elements of the other. A rigorous, efficient and scalable similarity measure is a pre-requisite of an ontology alignment process. This chapter presents a semantic similarity measure based on a matrix represention of nodes from an RDF labelled directed graph. An entity is described with respect to how it relates to other entities using N-dimensional vectors, being N the number of selected external predicates. We adapt the graph similarity calculation described in [20] when applying this idea to the alignment of two ontologies. We have successfully tested the model with the public testcases of the Ontology Alignment Evaluation Initiative 2005. [1]

10.1 Already published work

Large portions of this chapter have appeared in the following paper:

> Tous R., Delgado J. "A Vector Space Model for Semantic Similarity Calculation and OWL Ontology Alignment", 17th International Conference on Database and Expert Systems Applications (DEXA 2006), 4-8 September 2006. To be published in Lecture Notes in Computer Science.

[1] This work has been partly supported by the Spanish administration (DRM-MM project, TSI 2005-05277).

10.2 Introduction

10.2.1 Motivation

For many knowledge domains (biology, music, web directories, digital rights management, etc.) several overlapping ontologies (middle ontologies) are being engineered. Each one is a different abstraction and representation of the same or similar concepts. There are proliferating also a myriad of problem-specific ontologies (lower ontologies) for many applications, metadata repositories, personal information systems and peer-to-peer networks.

To enable collaboration within and across information domains, software agents require the semantic alignment (mapping) of the different formalisms. The alignment process will identify the equivalences between some entities (e.g. classes and properties) of the participating ontologies, and the different levels of confidence. These mappings are required before the querying of semantic data from autonomous sources can take place.

10.2.2 Ontology Alignment

Ontology alignment (or matching) is the operation that takes two ontologies and produces a set of semantic correspondences (usually semantic similarities) between some elements of one of them and some elements of the other. Several ontology alignment algorithms have been provided like GLUE [32], OLA [37] or FOAM [36]. A more formal definition, borrowed from [35], can be given:

Definition 10.2.1. Given two ontologies $\mathcal{O}$ and $\mathcal{O}'$, an *alignment* between $\mathcal{O}$ and $\mathcal{O}'$ is a set of correspondences (i.e., 4-uples): $< e, e', r, n >$ with $e \in \mathcal{O}$ and $e' \in \mathcal{O}'$ being the two matched entities, r being a relationship holding between e and e', and n expressing the level of confidence [0..1] in this correspondence.

It is typically assumed that the two ontologies are described within the same knowledge representation language (e.g. OWL [112]). Here we will focus on automatic and autonomous alignment, but other semi-automatic and interactive approaches exist.

10.2.3 Semantic similarity measures

The ontology alignment problem has an important background work in discrete mathematics for matching graphs [58][114], in databases for mapping schemas [122] and in machine learning for clustering structured objects [19]. Most part of ontology alignment algorithms are just focused on finding close entities (the "=" relationship), and rely on some *semantic similarity* measure.

A semantic similarity measure tries to find clues to deduce that two different data items correspond to the same information. Data items can be ontology classes and properties, but also instances or any other information representation entities. Semantic similarity between ontology entities (within the same ontology or between two different ones) may be defined in many different ways. The recently held Ontology Alignment Evaluation Initiative 2005 [108] has shown that the best alignment algorithms combine different similarity measures. [37] provides a classification (updating [122]) inherited from the study of similarity in relational schemas. This classification can be simplified to four categories when being applied to ontologies: Lexical, Topological, Extensional and Model-based.

10.2.4 Our approach

The work presented in this chapter takes a topological or structure-based semantic similarity approach. As ontologies and knowledge-representation languages evolve, more sophisticated structure-based similarity measures are required. In RDF graphs, relationships are labeled with predicate names, and trivial distance-based strategies cannot be applied. Some works like [64] explore similarity measures based on structure for RDF equivalent bipartite graphs.

Our work focus also in RDF, but faces directly the natural RDF labelled directed graphs. The approach can be outlined in the following two points:

1. To compute the semantic similarity of two entities we have taken the common RDF and OWL predicates as a semantic reference. Objects are described and compared depending on how they relate to other objects in terms of these predicates. We have modeled this idea as a simple vector space.
2. To efficiently apply our similarity measure to the ontology alignment problem we have adapted it to the graph matching algorithm of [20].

10.3 Representing RDF labelled directed graphs with a vector space model (VSM)

In linear algebra a *vector space* is a set V of *vectors* together with the operations of addition and scalar multiplication (and also with some natural constraints such as closure, associativity, and so on). A vector space model (VSM) is an algebraic model introduced a long time ago by Salton [134] in the information retrieval field. In a more general sense, a VSM allows to describe and compare objects using N-dimensional vectors. Each dimension corresponds to an orthogonal feature of the object (e.g. weight of certain term in a document).

In an OWL ontology, we will compare entities taking into consideration their relationships with all the other entities present in the ontology - First we will focus on similarity within the same ontology, next we will study its application to the alignment of two ontologies -. Because relationships can be of different nature we will model them with a vector space. For this vector space, we will take as dimensions any OWL, RDF Schema, or other external predicate (not ontology specific) e.g. *rdfs:subClassOf*, *rdfs:range* or *foaf:name.* We can formally define the relationship of two nodes in the model:

Definition 10.3.1. Given any pair of nodes n_1 and n_2 of a directed labelled RDF graph $\mathcal{G}_{\mathcal{O}}$ representing the OWL ontology $\mathcal{O}$, the relationship between them, $rel(n_1, n_2)$, is defined by the vector $\{arc(n_1, n_2, p_1), ..., arc(n_1, n_2, p_N)\}$, where arc is a function that returns 1 if there is an arc labelled with the predicate p_i from n_1 to n_2 or 0 otherwise. p_i is a predicate from the set of external predicates $\mathcal{P}$ (e.g. {*rdfs:subClassOf*, *foaf:name*}).

$$\begin{aligned} rel(n_1, n_2) = &\{arc(n_1, n_2, p_1), ..., arc(n_1, n_2, p_N)\} \mid \\ & n_1, n_2 \in \mathcal{G_O} \ \wedge \ \forall i \in [0; N], p_i \in \mathcal{P} \end{aligned}$$

$$arc(n_1, n_2, p_i) = \begin{cases} 1 & \text{if there is an arc labelled with } p_i \text{ from } n_1 \text{ to } n_2; \\ 0 & \text{otherwise.} \end{cases}$$

Example 10.3.1. Let us see a simple example. Take the following graph $\mathcal{G_A}$ representing an ontology $\mathcal{O_A}$. Imagine a trivial two-dimensional vector space to model the relationships between nodes. External predicates *rdfs:domain* and *rdfs:range* have been chosen for dimensions 0 and 1 respectively.

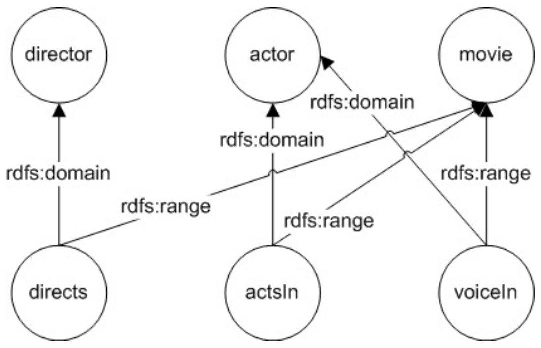


Figure 10.1: $\mathcal{G_A}$

The relationship between the property *directs* and the class *director* will be described by $\{1, 0\}$. The relationship between the property *actsIn* and the class *movie* will be described by $\{0, 1\}$, and so on.

Now, the full description of an entity can be achieved with a vector containing the relationships between it and all the other entities in the ontology. Putting all the vectors together we obtain a three-dimensional matrix $\mathcal{A}$ representation of the labelled directed graph $\mathcal{G_A}$ (row order: *director*, *actor*, *movie*, *directs*, *actsIn*, *voiceIn*):

$$\mathcal{A} = \begin{pmatrix} (0,0) & (0,0) & (0,0) & (0,0) & (0,0) & (0,0) \\ (0,0) & (0,0) & (0,0) & (0,0) & (0,0) & (0,0) \\ (0,0) & (0,0) & (0,0) & (0,0) & (0,0) & (0,0) \\ (1,0) & (0,0) & (0,1) & (0,0) & (0,0) & (0,0) \\ (0,0) & (1,0) & (0,1) & (0,0) & (0,0) & (0,0) \\ (0,0) & (1,0) & (0,1) & (0,0) & (0,0) & (0,0) \end{pmatrix}$$

10.4 Similarity of entities within the same ontology

In the general case, the correlation between two vectors x and y in an N-dimensional vector space can be calculated using the scalar product. We can normalize it by dividing this product by the product of the vector modules, obtaining the cosine distance, a traditional similarity measure. In our case, vectors describing entities in terms of other entities are composed by relationship vectors (so they are matrices). We can calculate the scalar product of two of such vectors of vectors V and W using also the scalar product to compute $V_i W_i$:

$$V \cdot W = \sum_{i=1}^{N} \sum_{j=1}^{M} V_{ij} W_{ij}$$

Applying this equation to the above example we can see that the scalar product of e.g. the vector describing *directs* and the vector describing *actsIn* is $directs \cdot \Delta actsIn = 1$. The scalar product of *actsIn* and *voiceIn* is $actsIn \cdot \Delta voiceIn = 2$, and so on. Normalizing these values (to keep them between 0 and 1) would allow to obtain a trivial similarity matrix of the ontology entities. However, we aim to propagate the structural similarities iteratively, and also to apply this idea to the alignment of two different ontologies. In the following sections we will describe how to do it by adapting the ideas described in [20].

10.5 Applying the model to an ontology alignment process

To calculate the alignment of two ontologies represented with our vector space model we have adapted the graph matching algorithm of [20]. This adapted algorithm calculates entity similarities in an RDF labelled directed graph by iteratively using the following updating equation:

Definition 10.5.1. $S_{k+1} = BS_kA^T + B^TS_kA, k = 0, 1, ...$
where S_k is the $N_B * N_A$ similarity matrix of entries s_{ij} at iteration k, and A and B are the $N_B * N_B * N_P$ and $N_A * N_A * N_P$ three-dimensional matrices representing $\mathcal{G_A}$ and $\mathcal{G_B}$ respectively. N_A and N_B are the number of rows of A and B, and P is the number of predicates selected as dimensions of the VSM.

Note that, as it is done in [20], initially the similarity matrix S_0 is set to 1 (assuming for the first iteration that all entities from $\mathcal{G_A}$ are equal to all entities in $\mathcal{G_B}$). If we start the process already knowing the similarity values of some pair of entities, we can modify this matrix accordingly, and keep the known values between iterations.

Example 10.5.1. Let's see a simple example. Take the following graphs $\mathcal{G_A}$ and $\mathcal{G_B}$. Figure 10.6 shows their corresponding RDF labelled directed graphs.

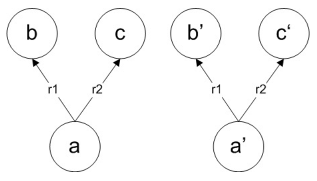


Figure 10.2: $\mathcal{G_A}$ (left) and $\mathcal{G_B}$ (right)

$$A = \begin{pmatrix} (0,0) & (1,0) & (0,1) \\ (0,0) & (0,0) & (0,0) \\ (0,0) & (0,0) & (0,0) \end{pmatrix} B = \begin{pmatrix} (0,0) & (1,0) & (0,1) \\ (0,0) & (0,0) & (0,0) \\ (0,0) & (0,0) & (0,0) \end{pmatrix} S_0 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

$$R_{ba} = BS_0 = \begin{pmatrix} (1,1) & (1,1) & (1,1) \\ (0,0) & (0,0) & (0,0) \\ (0,0) & (0,0) & (0,0) \end{pmatrix}$$

$$sim_{ba} = R_{ba}A^T = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$R_{ab} = S_0 A = \begin{pmatrix} (0,0) & (1,0) & (0,1) \\ (0,0) & (1,0) & (0,1) \\ (0,0) & (1,0) & (0,1) \end{pmatrix}$$

$$sim_{ab} = B^T R_{ab} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$S_1 = sim_{ba} + sim_{ab} = BS_0A^T + B^T S_0 A = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

To normalize the similarity matrix (to keep its values between 0 and 1) [20] divides all its elements by the Frobenius norm of the matrix, defined as the square root of the sum of the absolute squares of its elements [2].

$$S_1 = S_1/frobeniusNorm(S_1) = \begin{pmatrix} 0,816 & 0 & 0 \\ 0 & 0,408 & 0 \\ 0 & 0 & 0,408 \end{pmatrix}$$

Iterating the algorithm 4 times it converges to the following result:

$$S_4 = \begin{pmatrix} 0,577 & 0 & 0 \\ 0 & 0,577 & 0 \\ 0 & 0 & 0,577 \end{pmatrix}$$

So, as expected the entities a', b' and c' (rows) are similar to a, b and c (columns) respectively.

[2]Frobenius norm: $\sqrt{\sum_{i=1}^{M} \sum_{j=1}^{N} |a_{ij}|}$

10.5.1 Computational cost and optimization

Because the number of selected external predicates $p_i \in \mathcal{P}$ can be small and it is independent of the size of the ontologies, operations involving relationships vectors can be considered of constant cost, and the general algorithm of order $O(N^2)$. Because the number of nodes can be considerably high, some optimizations are required to constraint the processing time. Inspired in [64], we have classified nodes into five types: Properties (p), Classes (c), Instances (i), External Classes (c') and External Instances (i'). Because nodes from one type cannot be similar to nodes of another type, the matrices can be rewritten (rows and columns correspond to types previously mentioned and in the same order):

$$A = \begin{pmatrix} A_{p-p} & A_{p-c} & A_{p-i} & A_{p-c'} & A_{p-i'} \\ A_{c-p} & A_{c-c} & A_{c-i} & A_{c-c'} & A_{c-i'} \\ A_{i-p} & A_{i-c} & A_{i-i} & A_{i-c'} & A_{i-i'} \\ A_{c'-p} & A_{c'-c} & A_{c'-i} & A_{c'-c'} & A_{c'-i'} \\ A_{i'-p} & A_{i'-c} & A_{i'-i} & A_{i'-c'} & A_{i'-i'} \end{pmatrix}$$

$$S_k = \begin{pmatrix} S_p & 0 & 0 & 0 & 0 \\ 0 & S_c & 0 & 0 & 0 \\ 0 & 0 & S_i & 0 & 0 \\ 0 & 0 & 0 & S_{c'} & 0 \\ 0 & 0 & 0 & 0 & S_{i'} \end{pmatrix}$$

Definition 10.5.2. The S_{k+1} equation can be decomposed into three formulas:

$\mathbf{S_{p_{k+1}}} = B_{p-p}S_{p_k}A^T_{p-p} + B_{p-c}S_{c_k}A^T_{p-c} + B_{p-i}S_{i_k}A^T_{p-i} + B_{p-c'}S_{c'_k}A^T_{p-c'} + B_{p-i'}S_{i'_k}A^T_{p-i'} + B^T_{p-p}S_{p_k}A_{p-p} + B^T_{c-p}S_{c_k}A_{c-p} + B^T_{i-p}S_{i_k}A_{i-p} + B^T_{c'-p}S_{c'_k}A_{c'-p} + B^T_{i'-p}S_{i'_k}A_{i'-p}$

$\mathbf{S_{c_{k+1}}} = B_{c-p}S_{p_k}A^T_{c-p} + B_{c-c}S_{c_k}A^T_{c-c} + B_{c-i}S_{i_k}A^T_{c-i} + B_{c-c'}S_{c'_k}A^T_{c-c'} + B_{c-i'}S_{i'_k}A^T_{c-i'} + B^T_{p-c}S_{p_k}A_{p-c} + B^T_{c-c}S_{c_k}A_{c-c} + B^T_{i-c}S_{i_k}A_{i-c} + B^T_{c'-c}S_{c'_k}A_{c'-c} + B^T_{i'-c}S_{i'_k}A_{i'-c}$

$\mathbf{S_{i_{k+1}}} = B_{i-p}S_{p_k}A^T_{i-p} + B_{i-c}S_{c_k}A^T_{i-c} + B_{i-i}S_{i_k}A^T_{i-i} + B_{i-c'}S_{c'_k}A^T_{i-c'} + B_{i-i'}S_{i'_k}A^T_{i-i'} + B^T_{p-i}S_{p_k}A_{p-i} + B^T_{c-i}S_{c_k}A_{c-i} + B^T_{i-i}S_{i_k}A_{i-i} + B^T_{c'-i}S_{c'_k}A_{c'-i} + B^T_{i'-i}S_{i'_k}A_{i'-i}$

$S_{c'_{k+1}}$ and $S_{i'_{k+1}}$ are diagonal matrices passed as input parameters. They are kept unchanged between iterations.

10.5.2 Comparison against algoritms based on bipartite graphs

The use of an algorithm to measure similarity between directed graphs could lead to think that it would be better to directly apply it over the ontologies equivalent bipartite graphs (like it is done in [64]), instead of adapting it to RDF labelled directed graphs.

However, our approach has some advantages; on one hand we reduce critically the number of nodes and the computational cost. On the other hand, in bipartite graphs the core predicates of OWL are treated as all the other nodes, while in our model they become the semantic reference to describe and compare entities. Figure 10.7 shows the equivalent bipartite version of the previous example with two graphs of three nodes.

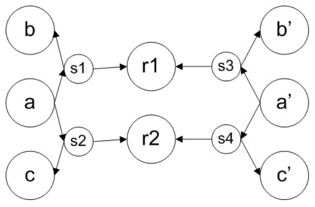


Figure 10.3: Bipartite version of Figure 10.6

Appying the [64] we obtain the following similarity matrix between a', b', c' and a, b, c:

Note that initially the similarity matrix X_0 is set to 1. If we start the process already knowing the similarity values of some pair of entities, we can modify set this matrix accordingly, and keep the known values between iterations. Let's calculate the similarity between $\mathcal{G}_{\mathcal{A}}$ and $\mathcal{G}_{\mathcal{B}}$:

$$A = \begin{pmatrix} 0&0&0&1&1&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 0&1&0&0&0&1&0\\ 0&0&1&0&0&0&1\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0 \end{pmatrix} B = \begin{pmatrix} 0&0&0&1&1&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 0&1&0&0&0&1&0\\ 0&0&1&0&0&0&1\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0 \end{pmatrix} X_0 = \begin{pmatrix} 1&1&1&1&1&1\\ 1&1&1&1&1&1\\ 1&1&1&1&1&1\\ 1&1&1&1&1&1\\ 1&1&1&1&1*&1\\ 1&1&1&1&1&1* \end{pmatrix}$$

Iterating the algorithm 22 times it converges to the following result:

$$X_{22} = \begin{pmatrix} 0,405&0&0&0&0&0&0\\ 0&0,153&0,05&0&0&0,153&0,05\\ 0&0,05&0,153&0&0&0,05&0,153\\ 0&0&0&0,534&0,172&0&0\\ 0&0&0&0,172&0,534&0&0\\ 0&0,153&0,05&0&0&1&0,05\\ 0&0,05&0,153&0&0&0,05&1 \end{pmatrix}$$

$$X_{22} = \begin{pmatrix} 0,405&0&0\\ 0&0,153&0,05\\ 0&0,05&0,153 \end{pmatrix}$$

As can be seen, the inclusion of statement nodes adds some symmetries not present in the original graphs, resulting in less precise results. Some similarities between nodes b' and c (and vice-versa) appear.

10.5.3 An extended example

Example 10.5.2. Let's see a simple example. Take the following graphs $\mathcal{G}_{\mathcal{A}}$ and $\mathcal{G}_{\mathcal{B}}$.

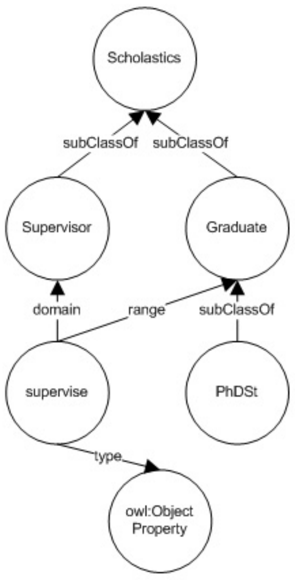


Figure 10.4: $\mathcal{G}_{\mathcal{A}}$

Iterating the algorithm 22 times it converges to the following result:

Rows: b:Teacher, b:OverseaStudent, b:People, b:Other, b:Student Columns: a:Graduate, a:Scholastics, a:PhdStudent, a:Supervisor

$$X_{12} = \begin{pmatrix} 0,049 & 0 & 0,014 & 0,106 \\ 0,013 & 0 & 0,02 & 0,013 \\ 0,051 & 0,125 & 0 & 0 \\ 0,018 & 0 & 0,014 & 0,018 \\ 0,145 & 0,029 & 0,014 & 0,049 \end{pmatrix}$$

Separately b:teach and a:supervise similarity = 0,446
After normalization:

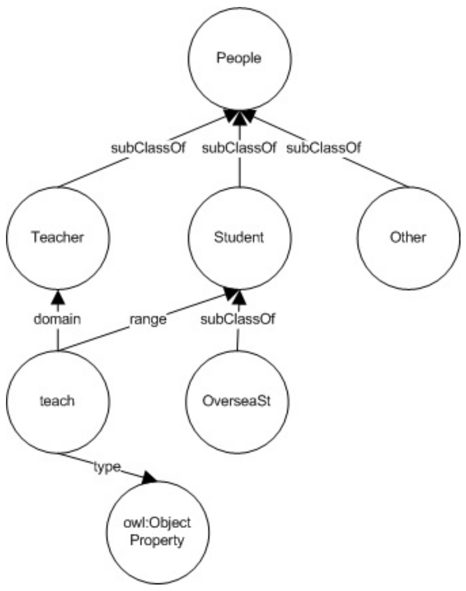


Figure 10.5: $\mathcal{G}_{\mathcal{B}}$

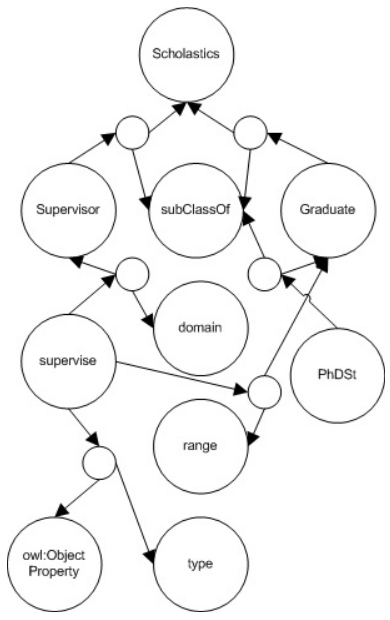


Figure 10.6: Bipartite version of $\mathcal{G}_{\mathcal{A}}$

$$X_{12} = X_1/maxValue(X_1) = \begin{pmatrix} 0,336 & 0 & 0,098 & 0,73 \\ 0,09 & 0 & 0,134 & 0,09 \\ 0,353 & 0,863 & 0 & 0 \\ 0,127 & 0 & 0,098 & 0,127 \\ 1 & 0,201 & 0,098 & 0,336 \end{pmatrix}$$

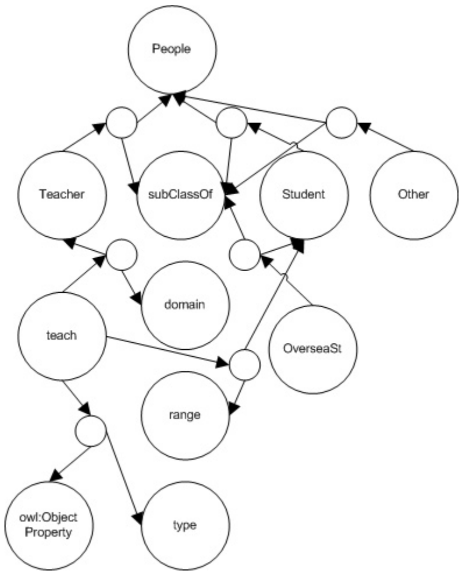


Figure 10.7: Bipartite version of $\mathcal{G}_{\mathcal{B}}$

Separately b:teach and a:supervise similarity = 1

So, as expected the entities a', b' and c' (rows) are similar to a, b and c (columns) respectively.

A Rows: a:PhdStudent, a:Graduate, a:Scholastics, a:Supervisor, a:supervise, owl:ObjectProper
B Rows: b:Other, b:People, b:OverseaStudent, b:Student, b:Teacher, b:teach, owl:ObjectProperty
Relationships: (subClassOf, domain, range, type)

$$A = \begin{pmatrix} (0,0,0,0) & (1,0,0,0) & (0,0,0,0) & (0,0,0,0) & (0,0,0,0) & (0,0,0,0) \\ (0,0,0,0) & (0,0,0,0) & (1,0,0,0) & (0,0,0,0) & (0,0,0,0) & (0,0,0,0) \\ (0,0,0,0) & (0,0,0,0) & (0,0,0,0) & (0,0,0,0) & (0,0,0,0) & (0,0,0,0) \\ (0,0,0,0) & (0,0,0,0) & (1,0,0,0) & (0,0,0,0) & (0,0,0,0) & (0,0,0,0) \\ (0,0,0,0) & (0,0,0,1) & (0,0,0,0) & (0,0,1,0) & (0,0,0,0) & (0,1,0,0) \\ (0,0,0,0) & (0,0,0,0) & (0,0,0,0) & (0,0,0,0) & (0,0,0,0) & (0,0,0,0) \end{pmatrix}$$

$$B = \begin{pmatrix} (0,0,0,0) & (1,0,0,0) & (0,0,0,0) & (0,0,0,0) & (0,0,0,0) & (0,0,0,0) & (0,0,0,0) \\ (0,0,0,0) & (0,0,0,0) & (0,0,0,0) & (0,0,0,0) & (0,0,0,0) & (0,0,0,0) & (0,0,0,0) \\ (0,0,0,0) & (0,0,0,0) & (0,0,0,0) & (1,0,0,0) & (0,0,0,0) & (0,0,0,0) & (0,0,0,0) \\ (0,0,0,0) & (1,0,0,0) & (0,0,0,0) & (0,0,0,0) & (0,0,0,0) & (0,0,0,0) & (0,0,0,0) \\ (0,0,0,0) & (1,0,0,0) & (0,0,0,0) & (0,0,0,0) & (0,0,0,0) & (0,0,0,0) & (0,0,0,0) \\ (0,0,0,0) & (0,0,0,0) & (0,0,0,0) & (0,0,0,1) & (0,0,1,0) & (0,0,0,0) & (0,1,0,0) \\ (0,0,0,0) & (0,0,0,0) & (0,0,0,0) & (0,0,0,0) & (0,0,0,0) & (0,0,0,0) & (0,0,0,0) \end{pmatrix}$$

$$X_0 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1* \end{pmatrix}$$

Iterating the algorithm 48 times it converges to the following result:

Rows: b:Other, b:People, b:OverseaStudent, b:Student, b:Teacher, b:teach, owl:ObjectProperty
Cols: a:PhD, a:Graduate, a:Scholastics, a:Supervisor, a:supervise, owl:ObjectProperty

$$X_{48} = \begin{pmatrix} 0 & 0,156 & 0 & 0,156 & 0 & 0 \\ 0 & 0 & 0,462 & 0 & 0 & 0 \\ 0,134 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0,399 & 0 & 0,156 & 0 & 0 \\ 0 & 0,156 & 0 & 0,354 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0,589 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

10.6 Results

To test our approach we have used the Ontology Alignment Evaluation Initiative 2005 testsuite [108]. The evaluation organizers provide a systematic benchmark test suite with pairs of ontologies to align as well as expected (human-based) results. The ontologies are described in OWL-DL and serialized in the RDF/XML format. The expected alignments are provided in a standard format expressed in RDF/XML and described in [108]. Because our model does not deal with lexical similarity, we have integrated our algorithm inside another hybrid aligner, Falcon [64] (replacing its structure similarity module by ours). This constraints the interest of the obtained results, but otherwise it hadn't been possible a comparative evaluation. Because most part of the tests include more lexical similarity than structural similarity challenges, our aligner and Falcon[3] obtain very similar results (the same for tests 101-104 and 301-304). The differences fall between tests 201-266, that we show in table 10.1.

Rows correspond to test numbers, while columns correspond to the obtained values of precision (the number of correct alignments found divided by the total number of alignments found) and recall (the number of correct alignments found divided by the total of expected alignments).

[3]A description of all the tests can be obtained from [108]. Our results for tests not present in the table are the same as those of Falcon, and can be obtained in [64]

	vsm		falcon		foam		ola	
test	prec.	rec.	prec.	rec.	prec.	rec.	prec.	rec.
205	0.90	0.89	0.88	0.87	0.89	0.73	0.43	0.42
209	0.88	0.87	0.86	0.86	0.78	0.58	0.43	0.42
230	0.97	0.96	0.94	1.0	0.94	1.0	0.95	0.97
248	0.83	0.80	0.84	0.82	0.89	0.51	0.59	0.46
252	0.64	0.64	0.67	0.67	0.67	0.35	0.59	0.52
257	0.66	0.66	0.70	0.64	1.0	0.64	0.25	0.21
260	0.44	0.42	0.52	0.48	0.75	0.31	0.26	0.17
261	0.45	0.42	0.50	0.48	0.63	0.30	0.14	0.09
262	1.0	0.27	0.89	0.24	0.78	0.21	0.20	0.06
265	0.44	0.42	0.48	0.45	0.75	0.31	0.22	0.14
266	0.45	0.42	0.50	0.48	0.67	0.36	0.14	0.09

Table 10.1: OAEI 2005 tests where our approach (vsm) obtains a different result than [64]

10.7 Related Work

The initial work around structure-based semantic similarity just focused on is-a constructs (taxonomies). Previous works like [90] measure the distance between the different nodes. The shorter the path from one node to another, the more similar they are. Given multiple paths, one takes the length of the shortest one. [148] finds the path length to the root node from the least common subsumer (LCS) of the two entities, which is the most specific entity they share as an ancestor. This value is scaled by the sum of the path lengths from the individual entities to the root. [89] finds the shortest path between two entities, and scales that value by the maximum path length in the is–a hierarchy in which they occur.

Recently, new works like [20] define more sophisticated topological similarity measures, based on graph matching from discrete mathematics. These new graph-based measures suit the particularities of the new ontologies, built with more expressive languages like OWL [112]. Our work is based on the previous work in [20], and also in its adaptation to OWL-DL ontologies alignment in [64]. This last work describes a structural similarity strategy called GMO (Graph Matching for Ontologies). Differently from our work, GMO operates over RDF bipartite graphs. It allows a more direct application of graph matching algorithms, but also increases the number of nodes and reduces scalability.

10.8 Conclusions

We have presented here an approach to structure-based semantic similarity measurement that can be directly applied to OWL ontologies modelled as RDF labelled directed graphs. The work is based on the intuitive idea that similarity of two entities can be defined in terms of how these two entities relate to the world they share (e.g. two red objects are similar with respect to the colour dimension, but their similarity cannot be determined in a general way). We describe and compare ontological objects in terms of how they relate to other objects. We model these relationships with a vector space of N dimensions being N the number of selected external predicates (e.g. *rdfs:subClassOf*, *rdfs:range* or *foaf:name*). We have adapted the graph matching algorithm of [20] to these idea to iteratively compute the similarities between two OWL ontologies. We have presented also an optimization of the algorithm to critically reduce its computational cost. The good results obtained in the tests performed over the Ontology Alignment Evaluation Initiative 2005 testsuite has proven the value of the approach in situations in which structural similarities exist.

Part III

Heterogeneous Query Interfaces: XML Query Tunneling

Chapter 11

Facing Heterogeneous Query Interfaces: Query Tunneling

In this chapter we describe the motivation, the requirements, the design decisions and some implementation aspects related to the development of an advanced metasearch strategy. This strategy is a reformulation of the traditional *answering queries using views* [51] problem in a *local-as-view* (LAV) scenario from the data integration discipline. We describe a new solution suited for the highly restricted and volatile scenario of the Web, and based on XML technologies.

11.1 Already published work

Large portions of this chapter have appeared in the following papers:

Gil R., Tous R., García R., Rodríguez E., Delgado J. "Managing Intellectual Property Rights in the WWW: Patterns and Semantics". 1st International Conference on Automated Production of Cross Media Content for Multi-channel Distribution (AXMEDIS 2005), November 2005

Tous R., Delgado, J. "Interoperability Adaptors for Distributed Information Search on the Web". Proceedings of the 7th ICCC/IFIP International Conference on Electronic Publishing 2003. http://elpub.scix.net/cgi-bin/works/Show?0341

Tous R., Delgado, J. "Advanced Meta-Search of News in the Web", Proceedings of the 6th International ICCC/IFIP Conference on Electronic Publishing. Publisher: VWF Berlin, 2002. ISBN 3-89700-357-0. 395 pages. http://elpub.scix.net/cgi-bin/works/Show?0234

11.2 Introduction

The main idea behind this work is that, in specific domains such as newspaper news, virtual libraries, videos or music repositories, the available metasearch engines usually offer very restricted user interfaces (often only a keywords field) because of the difficulty to face the interface particularities of each underlying source. Our approach targets to overcome this limitation with a strategy inspired in the old works related to data integration but making use of the new possibilities offered by XML technologies. We base our stratgy on the reprocessing of the metadata returned by the different sources, allowing to include metadata-based search conditions in the user interface even if they are not offered by some of the target engines.

11.2.1 Web search engines

When we are looking for some information or content, in any digital environment, we have two possible alternatives: We can explore one by one all the existing objects inside the set of interest that in the case of the Web could take probably more than a million years or we can use a search application that allows us to express constraints about the properties of the objects that we are seeking, using some kind of language as for example SQL in the context of databases. The more expressiveness the language has the more precision the query will have. Traditional search over the Internet is usually performed using applications known as 'search engines'. These systems seek a list of keywords among the textual content of the Web (HTML, PDF, etc.). The concordance of a resource with the query depends on the times keywords are present and also on their relative and absolute position.

The traditional search engines user interface consists of a single text field where users can enter a sequence of keywords and boolean operators to constraint how these keywords must be searched. Because common users are not programmers most of the search sites offer an "advanced search" page to facilitate an alternative way in boolean queries. Once the search is finished the search engine shows to the user a results page, where it lists the web resources where the keywords have been found. The list is showed in descendent order, from the best result to the worst according to the criteria described before. The items of the results list, in this kind of search, contain few information about the resource described: information about the title, a short description, the size and maybe the author.

11.2.2 Specialised search engines

In specific domains, as newspaper news, the available search engines use to offer to the users more complex interfaces than the generic ones. These interfaces allow to specify constraints about specific features of the resources being searched, as the date of an article, the price of a book, etc. The results page of a specialised search engine it is quite similar to the results page described in the previous section, but it provides more information about each item in the list.

Engines of different domains, as videos, music or games for example, will use a different set of attributes to describe each matching result, but even engines of the same domain, books in this case, will probably use a similar but not equal set of attributes. This is the main drawback that constraints the functionality of the existing specialised metasearch engines, as we will discuss later, and one of the targets of this part of the research work.

11.2.3 Metasearch engines

The increasing number of search engines has motivated the apparition of new systems that help users finding information in the Internet by automatically querying a set of available search engines. These systems are called metasearch engines. From the users point of view traditional meta-search engines have the same interface and functionality as normal ones, but it is commonly accepted that they are slower. Metasearch engines have to face the problem of querying applications designed for human interaction with interfaces as those described above. Some initiatives have appeared to define a standardized and machine-friendly access point to Web search systems, but the success of these approaches is constrained by the fact that search service providers are reluctant of other systems taking unrestrained profit of their work. However, the inexistence of machine-friendly interfaces cannot avoid the exploitation by third parties of the information harvesting effort of the existing search engines, mainly because they use browsers as a presentation layer, with exposed HTTP requests and HTML results pages. This leaves a door open to other applications to act as browsers and launch queries against them. So the task of meta-search can be divided in two main sub-problems:

1. How to query each search engine
2. How to obtain the information from each results page

The necessity to feature each engine interface, overall considering the lack of collaboration, is very time-consuming and cumbersome, and no one can guarantee that the interfaces will remain unchanged. This makes existing metasearch engines very difficult to maintain, and the uncertainness about their update state reduces their public acceptance.

11.2.4 Specialised Metasearch

If in the field of generic search we can find the figure of the meta-search engine, in the field of specialised search happens exactly the same. There exist some meta-search engines designed to launch queries against a set of specialised search engines of the same domain. Currently there exist specialised metasearch systems in practically every possible area. As said before, specialised search engines provide complex interfaces to perform accurate queries constraining the particular features of the target resources. Surprisingly, traditional specialised meta-search usually offer only a "one-field" interface to the user. The origin of this limitation lies in the difficulty to feature the interface particularities of every underlying specialised engine. To provide a richest interface it would be necessary to map the interface semantics with the semantics of every target engine, a hard task especially if we consider that the interfaces could change. To overcome this limitation is one of the targets of our research work concerning this area, as it will be further explained.

11.3 Our approach: Advanced metasearch

Our research group has been working during several years in practical solutions to improve the capabilities of metasearch engines [118]. It is not a quantitative approach, since we do not pretend to reduce the search time or to amplify the search field, but a qualitative approach. The target is to provide users (human or agents) with a search interface that allows to express unrestricted search criteria over any existing resource in the Internet, without doing any presumption over the existing technologies (as other approaches do, as the Open Archives Initiative [109]). To achieve these goals we have defined a new strategy to design meta-search engines. This strategy is based on five main ideas:

1. Common metadata specification: The specification of a selected common set of properties (metadata) of the objects targeted by the search. This specification could be formalised using XML DTDs, XML Schemas or RDF Schemas for example.

2. User-interface independent query language: The specification of a generic query language that will be the entry point to the meta-search engine. It is not necessary to reinvent the wheel, if we assume that results will come in some XML form, W3C's XQuery [154] language will suffice (or RQL if we are using RDF). The language needn't to be known by human users because it could be distilled from human-friendly interfaces.

3. Human/machine maintainable XML descriptors: The use of XML descriptors to feature the 'hostile' underlying engines interfaces, to facilitate its generation and maintainability by human administrators or learning agents.

4. Mapping: The XML descriptors should allow to map the generic queries of the users (formalized in the language mentioned above) to the specific interfaces of the underlying engines. These descriptors should also be used to map the heterogeneous results obtained to the generic set of metadata. The homogeneous results obtained could be formalized using XML or RDF. Some questions arise here, as what happens with search conditions that cannot be mapped to some engines, or what must be done with results where not all the properties were defined (specially the properties referenced in some of the search conditions). The following point will answer these questions.

5. Reprocessing: The key aspect of our strategy is the reprocessing of the results. Because some of the conditions expressed by the generic user query probably cannot be mapped to all the underlying engines, it is necessary to reprocess the query over the obtained results, once they have been normalised. Because the user query arrives to the system in the form of a standard query language (XQuery, RQL, etc.) this stage can be performed by simply executing the respective query processor over the obtained results. This step guarantees that the results returned to the user are coherent with the conditions expressed in the initial query.

Our approach can be applied to any kind of search over the Web, but it becomes specially appropriate when it is applied to specialised meta-search. The reason is that, in despite of that the specialised search engines of the same domain use to share similar and rich sets of metadata, the traditional specialised meta-search has not found till now a way to exploit it, unless by establishing partnerships and specific protocols with the underlying engines administrators. Fig. 11.1 illustrates graphically the main features of this strategy, that leaves undefined some points marked here with dotted lines.

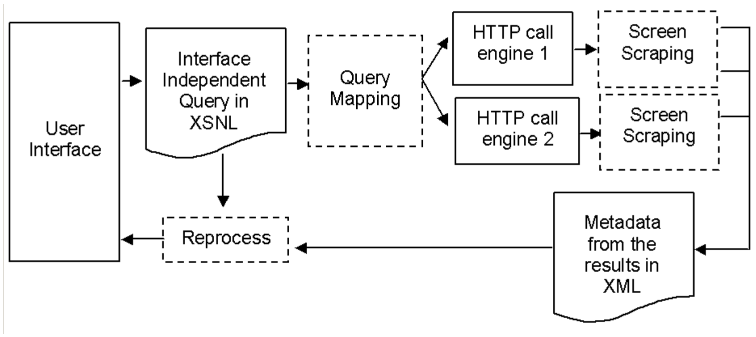


Figure 11.1: Strategy Diagram

11.4 XML Search Neutral Language (XSNL)

Any query, expressed in some syntax over a set of items featured with a certain set of properties, has an underlying semantic. Unfortunately, there is not an universally accepted standard way to express it. The parameters of the HTTP forms from Web search engines can be viewed as the building blocks of a particular query syntax, one for each different engine. The problem to map a generic user query, expressed in some query language, to a set of different HTTP interfaces is the problem of mapping between query syntaxes.

Imagine two news search engines; one offers the possibility to constraint the search to news appeared today, this week and this year. The other engine allows to constraint the search to an explicit month from the three last years. These are two different syntaxes. If we would have a syntax that allowed to express a specific range of dates, we could map it to each one of the syntaxes. So we need a language that acts between the meta-search interface and the target systems, built to be as flexible and fine grained as possible, allowing to map the biggest set of possible query conditions.

The old research paper (1995) *"Answering Queries Using Views"* [55] by A. Halevy et al. faces a similar situation and considers the problem of rewriting a conjunctive query using a set of conjunctive views. As most part of similar works of these initial approaches it uses Datalog[1]. Instead of Datalog, we have chosen XML related technologies as a more natural way to interact with modern web interfaces.

We have defined an XML-based query language to test our approach. We call it XML Search Neutral Language (XSNL) and we have applied it to the development of an advanced meta-search engine specialized in newspaper news, as explained in the next sections. The reason why we do not use XML Query as the intermediate language (we use it to process XSNL sentences) is that our strategy is based on having simple queries expressed in XML. There exists also an XML serialization of XML Query, but it is too verbose and complex to suit our approach. This doesn't mean that users (developers) cannot use XML Query to process the results, because XSNL is just used as a mediator query language with XML output.

The following XML code shows a sample instance of XSNL in the news context:

[1]see the *Background Information* chapters for a brief introduction to Datalog

```
XSNL  sample query:

<query>
  <select>
    <property propertyname="headline" />
    <property propertyname="date"  />
  </select>
  <where>
    <contains propertyname="content" value="iran" />
    <between  propertyname="date" from="2002-06-31" to="2003-06-31" />
    <in minimum="3" propertyname="source">
      <valueitem value="El Mundo" />
      <valueitem value="ABC" />
      <valueitem value="La Vanguardia" />
      <valueitem value="El Pais" />
      <valueitem value="Reuters" />
      <valueitem value="CNN Spain" />
      <valueitem value="Le Monde" />
      <valueitem value="The Washington Post" />
      <valueitem value="BBC" />
      <valueitem value="Diari Avui" />
      <valueitem value="La Stampa" />
    </in>
  </where>
  <sortby propertyname="source_order" type="asc"/>
  <sortby propertyname="date"/>
</query>
```

To understand the example, let's imagine a web page where a user can search newspaper news by specifying some keywords and a date range. In the web server the user request is analysed and translated to XSNL. Finally the XSNL is sent to the metasearch engine and the search process begins. The structure of a XSNL document is inspired in the SQL language. It have a 'select' element, where can be specified the desired attributes of the resulting objects, a 'where' element, where can be specified the search constraints, and an 'sortby' element, to determine the results order. The different constraints can be specified by using different elements, allowing to add to the language new constraint types with different structures.

11.5 A Practical Application: Advanced News Meta-search Engine

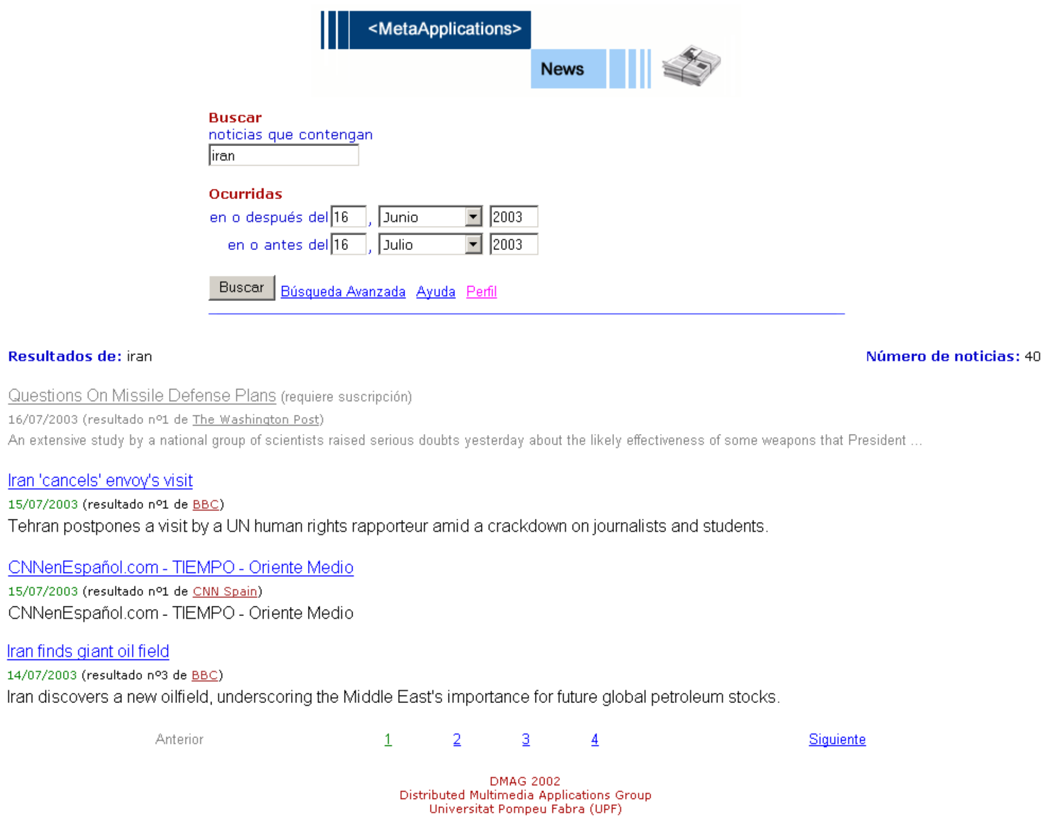
We have applied our ideas in the development of an advanced metasearch engine specialised in newspaper news [144]. In this domain there exist thousands of commercial and non-commercial traditional search engines, and also hundreds of available meta-search applications. The most part of the newspapers with presence in the Web offer search services in their sites. All these engines are the potential information sources of our system, but each one of them uses a different set of parameters in the queries and a different results page format. Our objective is to offer to the user a generic interface that allows to specify unrestricted conditions over the set of common properties that we have selected in this domain (headline, author, date, section, page, newspaper and language). To achieve this target, once selected and formalized (we are currently using a XML DTD) the subset of metadata of interest, the next step is to analyse the interface of every engine to obtain information about the query method. We use XML descriptors to describe how to map each specific set of query parameters to the generic common properties selected. We plan to use learning agents to perform this operation periodically because the interfaces of the engines could change over time. As a part of the interface featuring we must also acquire information about the results page, that will be used during the parsing process. Once we have a mechanism to feature the engines interfaces, we can design the interface of the meta-search engine. We have selected XML messages (SOAP [136]) containing XQuery sentences and HTTP protocol. This interface is open and can be used by third-parties to develop independent clients -user interfaces or agents- However, to demonstrate the functionality of the system, we have developed our own interface (see fig. 11.2 or fig. 11.3 for the advanced interface). The criteria specified by the user is translated to XQuery and sent to the metasearch engine. The engine maps the parts of the query (at least those that are possible) to each underlying engine interface and then launches all the searches in parallel. The results obtained are heterogeneous and must be parsed and mapped to the common set of properties. Because no one can guarantee that all the criteria have been mapped to all the engines, the results (now homogeneous and serialized in XML) must be reprocessed. This reprocessing is easily performed in the server only by using a XQuery processor with the XQuery received as the input.

11.6 Implementation

Implementing the prototype application has meant to instantiate the metasearch strategy explained above. The next subsections explain how, with the help of W3C's XML Query Language [154], the prototype executes the following query over the different sources:

```
XSNL  sample query:

<query>
  <select>
    <property propertyname="headline" />
    <property propertyname="date"  />
  </select>
  <where>
    <contains propertyname="content" value="test" />
    <between  propertyname="date" from="20020631" to="20030631" />
  </where>
```



Buscar
noticias que contengan
iran

Ocurridas
en o después del 16 , Junio 2003
en o antes del 16 , Julio 2003

Buscar Búsqueda Avanzada Ayuda Perfil

Resultados de: iran **Número de noticias:** 40

Questions On Missile Defense Plans (requiere suscripción)
16/07/2003 (resultado nº1 de The Washington Post)
An extensive study by a national group of scientists raised serious doubts yesterday about the likely effectiveness of some weapons that President ...

Iran 'cancels' envoy's visit
15/07/2003 (resultado nº1 de BBC)
Tehran postpones a visit by a UN human rights rapporteur amid a crackdown on journalists and students.

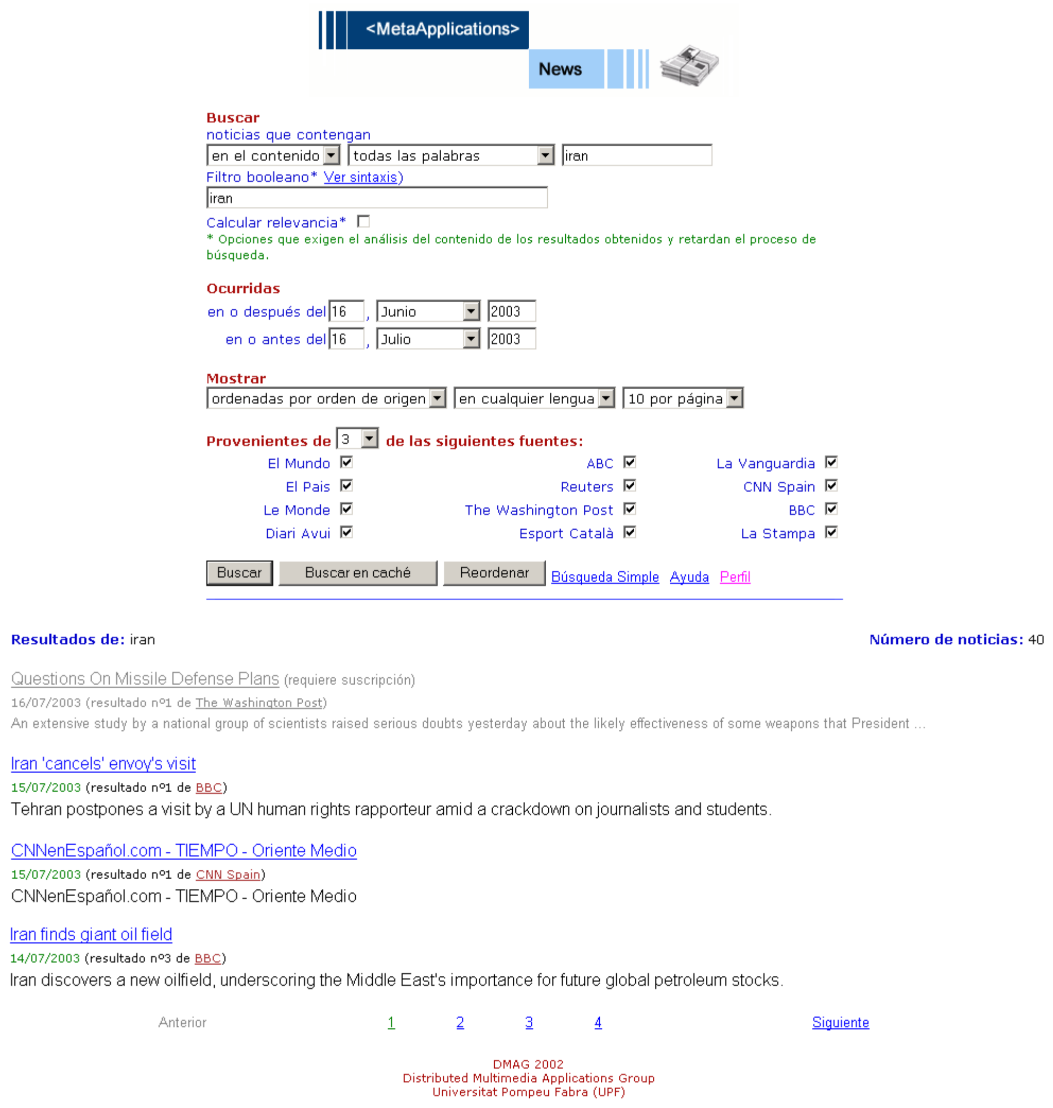
CNNenEspañol.com - TIEMPO - Oriente Medio
15/07/2003 (resultado nº1 de CNN Spain)
CNNenEspañol.com - TIEMPO - Oriente Medio

Iran finds giant oil field
14/07/2003 (resultado nº3 de BBC)
Iran discovers a new oilfield, underscoring the Middle East's importance for future global petroleum stocks.

Anterior 1 2 3 4 Siguiente

DMAG 2002
Distributed Multimedia Applications Group
Universitat Pompeu Fabra (UPF)

Figure 11.2: DMAG's News Advanced Meta-search Engine user interface



Buscar
noticias que contengan
en el contenido | todas las palabras | iran
Filtro booleano* Ver sintaxis)
iran
Calcular relevancia* ☐
* Opciones que exigen el análisis del contenido de los resultados obtenidos y retardan el proceso de búsqueda.

Ocurridas
en o después del 16 , Junio 2003
en o antes del 16 , Julio 2003

Mostrar
ordenadas por orden de origen | en cualquier lengua | 10 por página

Provenientes de 3 **de las siguientes fuentes:**

El Mundo ✓	ABC ✓	La Vanguardia ✓
El Pais ✓	Reuters ✓	CNN Spain ✓
Le Monde ✓	The Washington Post ✓	BBC ✓
Diari Avui ✓	Esport Català ✓	La Stampa ✓

Resultados de: iran **Número de noticias:** 40

Questions On Missile Defense Plans (requiere suscripción)
16/07/2003 (resultado nº1 de The Washington Post)
An extensive study by a national group of scientists raised serious doubts yesterday about the likely effectiveness of some weapons that President ...

Iran 'cancels' envoy's visit
15/07/2003 (resultado nº1 de BBC)
Tehran postpones a visit by a UN human rights rapporteur amid a crackdown on journalists and students.

CNNenEspañol.com - TIEMPO - Oriente Medio
15/07/2003 (resultado nº1 de CNN Spain)
CNNenEspañol.com - TIEMPO - Oriente Medio

Iran finds giant oil field
14/07/2003 (resultado nº3 de BBC)
Iran discovers a new oilfield, underscoring the Middle East's importance for future global petroleum stocks.

Anterior 1 2 3 4 Siguiente

DMAG 2002
Distributed Multimedia Applications Group
Universitat Pompeu Fabra (UPF)

Figure 11.3: DMAG's News Advanced Meta-search Engine advanced user interface

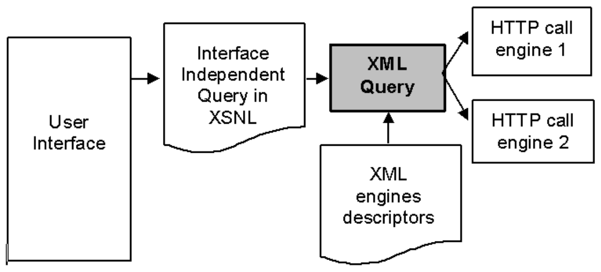


Figure 11.4: Mapping XSNL to each engine with XML Query

```
  <sortby propertyname="date" />
</query>
```

11.6.1 Mapping the user query to target systems

The first hole to fill from the strategy diagram described above is the way to obtain the parameters for the HTTP call for each search engine from the user query formalised in XSNL. This is not a trivial issue, because it must be determined if the condition expressed in one parameter is also expressed with one ore more statements in the XSNL query, and then extract the necessary information to give a value to the parameter. This process can require a complex analysis of the query, and the information to do it must not be coupled with code, for maintainability reasons. The description of the way to characterize a parameter of one engine should be editable, human-friendly and modifiable at runtime. Here is where the XML Query language fits, as is illus-trated in fig. 11.4. Each engine parameter is featured in XML. The parameters that require some analysis of the user query are featured with a XML Query within a CDATA clause. The following simplified example shows a fragment of the configuration file of a news metasearch engine.

```
Configuration file fragment:

<parameter>
  <name>precision</name>
  <type>xquery</type>
  <value>
    <![CDATA[
      <result-value>
        LET $a := document('input.xml')
        LET $b := $a//contains/@type
        LET $c := IF ($b = 'or') THEN '1'
                  ELSE IF ($b = 'and') THEN '2'
                  ELSE '3'
        RETURN $c
      </result-value>
    ]]>
```

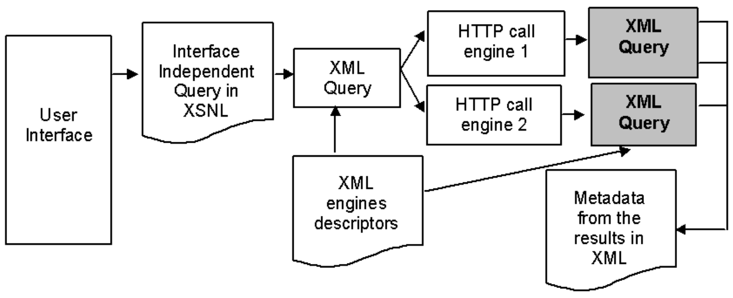


Figure 11.5: Screen Scraping with XML Query

```
  </value>
  <default>1</default>
</parameter>
```

The parameter *precision* of the target engine corresponds to a the boolean operator to be applied. This information is obtained from the user XML query, from the attribute 'type' of the 'contains' element. This is a very simplified example, but it serves to illustrate the idea.

11.6.2 Metadata extraction ("Screen Scraping")

The second hole to fill in the design of our system is the way the results pages are analyzed to extract the metadata related to each item. As we will see later, it would be convenient to formalize this metadata in XML. So, why not to convert the HTML results page in XML and then apply a XML Query to it? There exist a lot of tools to convert a HTML page to XML, even if it is malformed, and some of them perform very fast. An example is W3C's HTML Tidy [143]. The XML Query can be edited and modified by human administrators or software agents at runtime, without the necessity to recompile the sources. The following XML fragment shows an example on how to apply this idea to extract information from the results pages of the Washington Post searcher:

Example of XML query wrapper for screen scraping:

```
<resultsmap>
  <![CDATA[
    import dt as org.dmag.metasearch.utils.XQueryTransformDate;
    import ps as org.dmag.metasearch.utils.XQueryTransformString;
    FOR $c IN document('input.xml')/table
    WHERE $c/tr/td/font/b/a
    RETURN
    <result>
      <property name='headline' value=$c/tr/td[1] />,
      <property name='description' value=$c/tr/td[2] />,
      <property name='date' value=dt(ps($c/tr/td[3])) />,
```

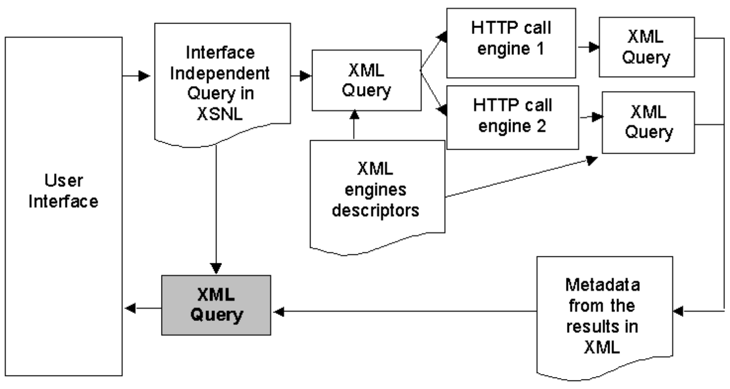


Figure 11.6: Reprocessing of the results with XML Query

```
      <property name='link' value=$c/tr/td[1]/a/@ref/text() />,
    </result>
   </results>
  ]]>
</resultsmap>
```

11.6.3 Reprocessing the results

Once we have obtained the results in XML, we must face the problem of reprocessing them to assure that all the initial conditions have been applied. The good news is that XML Query fits perfectly to do this job, the bad news is that our initial query is formalized in XSNL, not in XML Query. To overcome this problem we must simply transform our instance of XSNL to a XML Query, as it is shown in the following XML fragment:

```
<results>
    for $c in document('input.xml')//result
    where $c/property[@name='date']/@value .>=. '20020631'
      and  $c/property[@name='20030631']/@value .<=. '20030631'
  return $c
  sortby (property[@name='date']/@value
</results>
```

Now we already have a XML Query and we can just apply it to the results. The reprocessed results will be the output of the system, being the interface is responsible to render it in a convenient way. With this we have completed the initial strategy filling all the undefined aspects, as presented in fig. 11.6.

11.7 Related work

In the *State of the Art* chapters we have seen that the problem faced in this work is traditionally known as *the querying problem* of the data integration discipline. This problem is related to the ability to reformulate a query to combine information from the different sources according to their relationships with a mediated schema. This virtual mediated schema of data is the only schema visible to the users and their queries, giving them the illusion of interacting with one single information system.

There are two classic approaches concerning mediated schemas, the *global-as-view* (GAV) [2][26][50], which defines the mediated schema as a set of views over the data sources, and the *local-as-view* (LAV) [98][56][34], which takes the inverse point-of-view and describes sources as views over the mediated schema. Our approach is similar to the LAV scenario, because describes sources in terms of a mediated schema, and offers a very simplified solution to the problem of *answering queries using views.*

Traditional works on classic metasearch are [33] [135]. A more recent approach very similar to ours can be found in [81], that defines this idea as "Query Tunneling". It makes use of RDF tools and focuses on scientific papers databases.

11.8 Conclusions

Nowadays the Web has become the first place where people goes when they need to find some information. Surprisingly, and in concordance with what we have exposed in this document, we can affirm that the functionality of the current search systems of the Web is very limited, overall in comparison with other digital environments. Today, the 'keywords paradigm' consisting in that one types some words in a text field and press the 'search' button, satisfies the necessities of the most part of the people, and probably the average Web user does not want to hear nothing about new search interfaces. However, the Web is growing exponentially, and also the need for information, and soon the results obtained from a query based on a list of keywords will be unmanageable, and new and faster search mechanisms will be needed. Furthermore, in the short term, the most part of the non-textual resources of the Web (images, videos, music, etc.) will be enriched with some kind of metadata, supporting new standards as MPEG-7 [101]. The queries targeting these resources must be capable to express complex conditions about properties and attributes. In the long term, the 'Semantic Web' will require strategies to adapt the existing human-oriented search services to enable its use by software agents without traumatic impact in the underlying technologies. Our approach targets all these challenges without making assumptions of the success of some standard or protocol.

Chapter 12

Waiting Policies for Distributed Information Retrieval on the Web

Distributed Web search engines, those systems that query on-the-fly a set of available Web search systems in response to a user's request, have to face the problem of interacting with a large set of unpredictable systems with heterogeneous and changing response times. The necessity to achieve a compromise between the final user perceived delay and the quality of the results motivate the discussion between different algorithms to determine when the query process of an information source should be aborted. We call these algorithms 'waiting policies', and their goal is to maximise the quality of service (QoS) by minimizing the impact of sources behaviour without reducing the result quality beyond user's tolerance. Our experience in the design and development of specialised metasearch engines has given us the possibility to try some of these policies, and to extract some conclusions that we present here.

12.1 Motivation

A distributed search engine is a system that sends queries to multiple search systems, then collates the results in some way and formats them for display. We can identify a lot of different kinds of these systems, depending on their data sources, that can be internal indexes, associated text search engines, database search engines, message archives, Intranet or Web wide search engines, or even file servers.

This part of the work focuses on distributed search over Web search systems, in any of its forms (metasearch, content syndication, aggregation, etc.), but mainly over specialized ones. The typical session when using a traditional distributed Web search engine begins when the user submits a query to the system through the user interface. The engine then sends the user query to a set of underlying search engines (or component search engines [146]). The query must be translated to an appropriate format for each local system. Once received the results from the underlying sources, these are merged into a single ranked list and presented to the user. Nevertheless, we are not going to discuss here the motivation or interest of metasearching or other forms of distributed search on the Web, already well documented (see for e.g. [135], [33] or [88]).

Generally, and independently of the kind of systems the distributed engine is targeting, the process of querying the different engines is done in parallel. Each target engine is called and treated by a different execution thread of the distributed engine, that waits until the source (i.e. the queried system) responds with a set of results and then passes the information to a results pool. A main thread is responsible to analyse the evolution of the results pool and determine when the process must end. The set of rules used to do this is that we call the 'waiting policy', and it is the focus of this article.

Any distributed search system needs a waiting policy, but depending on the context the complexity of the policy can vary. If we have a system that queries a small set of remote sources and requires all the responses to fulfil the process, the waiting policy will consist in simply determining if some of the remote systems are out of service (even this can be a non trivial issue). This is what happens with most of traditional Web-wide meta-search systems (for e.g. [135]). However, if we have a Web-based distributed search engine that queries hundreds or even thousands of sources, we can consider that maybe it is not necessary nor desirable to wait them all (even if they are on service).

Maybe the user prefers to sacrifice some results to achieve a better response time. This also may happen when working with specially unreliable underlying sources, as in some specialised domains. But, how to measure the 'quality' of the results? And, when the 'quality' is enough to decide to abort the search process? These are difficult questions that depend on subjective variables like user's perception of the results relevance or the response time. We have been forced to study this problem as a side effect of our work on an advanced metasearch strategy [118], specially for the development of the architecture that instantiates this strategy and that is being used in real engines like [144].

12.2 Distributed Search Engine Performance

Some of the measures proposed to quantitatively measure the performance of classical information retrieval systems (see, e.g., [96]) can be extended to evaluate Web search and distributed search engines. However, as remarked by [85] Web users may have a tendency to favour some performance issues more strongly than traditional users of information retrieval systems. For example, interactive response times appear to be at the top of the list of important issues for Web users. A basic model from traditional retrieval systems [145] recognizes a three way trade-off between the speed of information retrieval, precision and recall (see fig. 12.1).Precision is the ratio of relevant documents to the number of retrieved documents:

$$precision = \frac{relevantDocuments}{retrievedDocuments} \tag{12.1}$$

Recall is defined as the proportion of relevant documents that are retrieved with respect the total number if existing relevant documents:

$$recall = \frac{relevantDocuments}{totalRelevantDocuments} \tag{12.2}$$

The precision is related to the expressiveness of the queries and the structure of the information to explore. Most Web users are not so much interested in the traditional measure

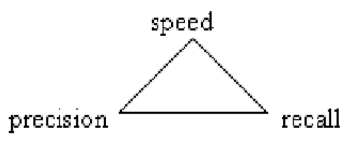


Figure 12.1: Three way trade-off in search engine performance [85]

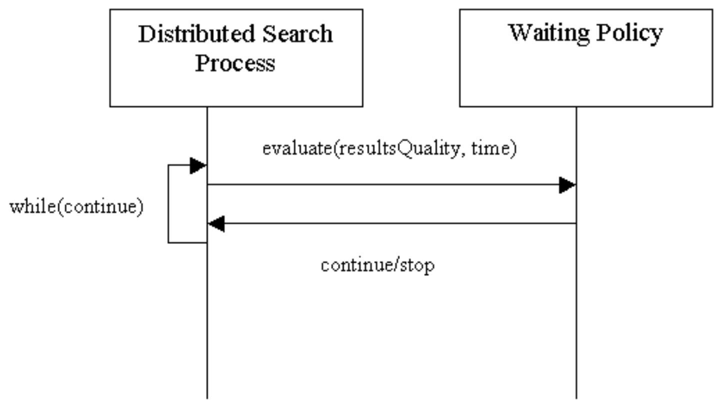


Figure 12.2: Waiting Policy

of precision and recall as the precision of the results displayed in the first page of the list of retrieved documents, before "next page" command is used. This special characteristic can be extended to meta-search engines users, and is a key point when considering to design a waiting policy, because it shows that users may prefer to sacrifice some of the results (even relevant ones) to improve response time.

12.3 Waiting Policy

A waiting policy (see fig. 12.2) is an algorithm that determines when a distributed information search process must end. A distributed search implies the interaction with remote, heterogeneous and potentially unreliable systems, with different and variable response times. This does not imply only that some engines can sporadically appear out-of-service, but this can also imply that some engines may experience enormous delays sometimes unrelated to network overloads. This situations can last hours, days or even weeks. So it is not enough to activate a method to detect source failures, because some sources can be on-service but with response times beyond user tolerance. Because this is a very dynamic context, the system must have a mechanism to determine when the search process must be stopped.

The algorithm must guarantee the quality of service (QoS) by optimising the relationship between the 'results quality' and the user's perceived response time. The response time can be easily measured but the results quality depends on a combination of objective

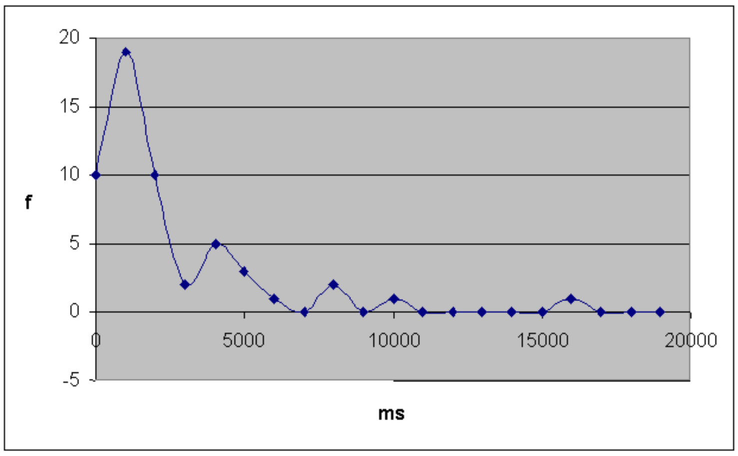


Figure 12.3: Distribution of Engines Delay

and subjective aspects. Objective aspects can be the correctness of the results in relation to the query, the total number of results or the number of sources successfully queried. The policies that we will study here focus on the third of these aspects, the number of sources, that has a tight relationship with results quality. Be-cause the quality of the sources can vary, even depending on the user perception, one can establish pondering mechanisms, assigning different weights to each source of information. However we will talk about 'number of successful sources' assuming that, if weights have been assigned, the number already reflects it.

12.4 Target Engines Behaviour

When designing a waiting policy, and taking the number of successful sources as a main parameter, it is interesting to know if there is some pattern in the behaviour of the target engines. We have tested the response times of an arbitrary set of approximately sixty search engines[1] (see fig. 12.3). The horizontal axis (fig. 12.3) represents the sequence of delay times and the vertical axis the number of target engines. The measurements have been grouped in intervals of 1000 ms (the graph shows the lower margin of the interval, for e.g. the first ten engines finished in less than one second). Knowing this we can anticipate how the results pool will evolve, because when each engine terminates, the pool receives a new result[2]. So, the function that models the time evolution of the number of results first grows slowly, because only a few set of engines have very small delay times, then grows very fast, and finally grows slowly again (see fig. 12.4). To construct the graph we have made a one-to-one association between results and engines, but the conclusions can be extended to the situation that we have mentioned before, when we assign different weights to the

[1]We have chosen about sixty heterogeneous engines from diverse locations. They include Web-wide search engines, specialized search engines, meta-search engines, digital libraries and others. The complete list can be download at http://www.tecn.upf.es/˜`ertous/projects/ir/url_waiting_policies.htm`

[2]Here for 'result' we mean a set of results returned from one source

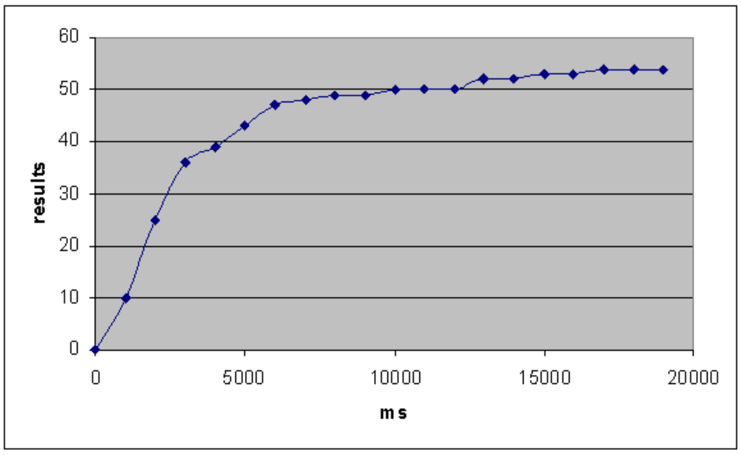


Figure 12.4: Results Evolution

sources.

Because target engines are remote systems arbitrarily located, we do not know the reasons that underlay their response time. Despite that, they are all conditioned by global changes of network conditions, overall considering that the response time includes the results transmission to the distributed search engine, around which they share the same network state. So, we can assume that target engines will have a certain global behaviour, and that this behaviour can change. Fig. 12.5 shows the progression of the results obtained from the same engines of the previous figures but under different network conditions. We have simulated a network overload in the proximities of the distributed search engine that have increased the delay of all the sources.

12.5 Results vs. Time

The simplest waiting policy one can imagine is a fixed Timeout Policy. Someone fixes heuristically a time limit for waiting for results; once surpassed, engines that still have not responded are discarded. In this case, users always experience the same delay (without considering the time it takes to send collated results to them). When network conditions are bad, only a few set of engines have the chance to respond, and users get only a small subset of the potential results. Obviously this cannot be considered a good policy but serves to illustrate what is the problem we are facing. The only advantage of a Timeout policy is that does not propagate the changes of target engines behaviour to end users, as illustrated in fig. 12.6. The figure shows how a degradation of network conditions slows down the harvesting of results that ends when time-out arrives (vertical line).

The opposite approach to Timeout Policy is the Minimum-results Policy, that waits until a minimum number of engines have successfully finished. This policy guarantees the quality of the response, but propagates to the user the delay time of the target engines and its changes (see Fig. 12.7). Again it must be mentioned here that we focus on the number of successful engines to measure the results quality assuming that this value could have been

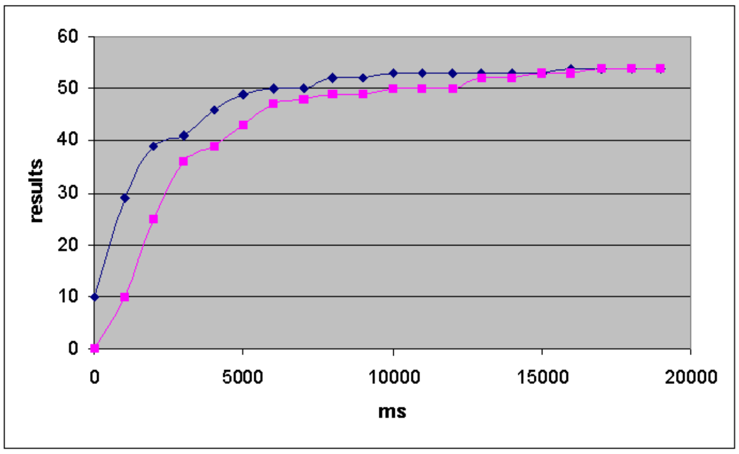


Figure 12.5: Results progression in hostile network conditions

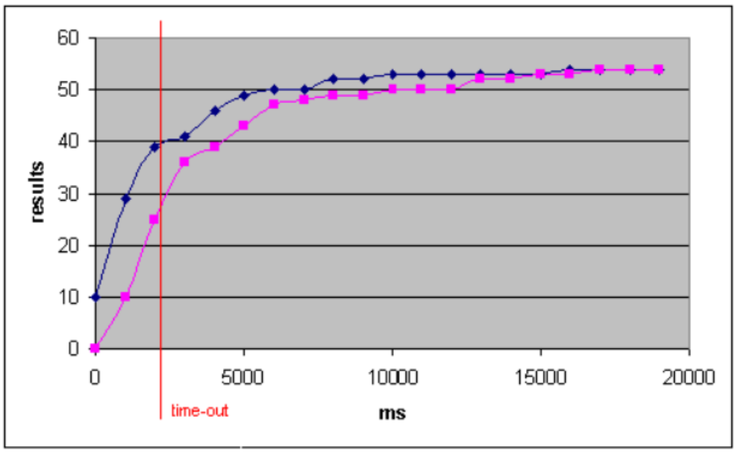


Figure 12.6: Timeout Policy Example

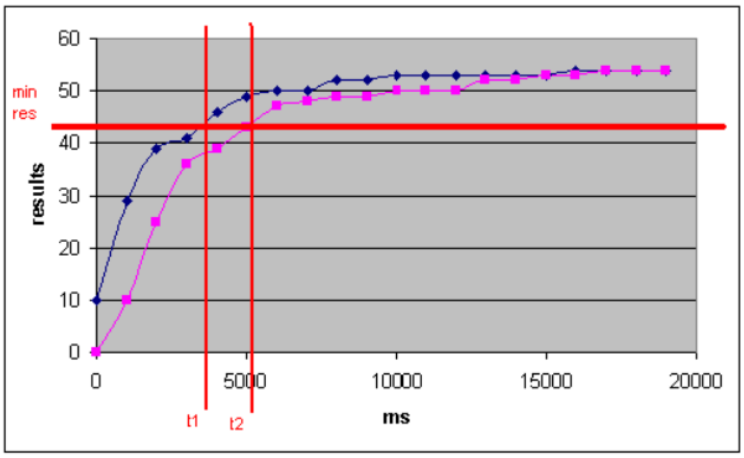


Figure 12.7: Minimum-results Policy Example

modified to reflect source relevance issues.

These two different approaches optimise only one of the variables, the response time or the number of successful sources respectively. The user experience with a system instantiating one of these algorithms can be very frustrating. In hostile network conditions the timeout policy may abort practically all search threads, eliminating the possibility to find answers to even simple queries. If network performance is good, the minimum-results policy will abort threads even with a very fast response time. It is clear that none of these algorithms is the solution, because it should optimise the two variables. To design a good waiting policy we should achieve a compromise between time and quality, and this must be able to adapt to environmental changes.

12.6 Source Discarding Policies

Some policies focus on being able to detect target failures under changing network conditions. These policies try to maximise the number of results without discarding any potential source. One approximation to this is to maintain statistics about each source delay, and, when network conditions vary, be able to determine the expected delay for each target. If we have N sources and K previous search experiences, we can trivially calculate the average delay (d) of each source to represent its historical performance (see Fig. 12.8). Because we probably want to use the latest information (the information of the search process in course), in the search experience K+1 we can approximate the future delay of some engine e applying the average change ratio of the n sources already finished.

This kind of policies work fine to discard sources that are behaving disaccording to the network conditions and its own historical values. However these policies propagate to the user the changes in the network performance, and cannot be considered a definite solutions according to what we have said before.

$$\bar{d}_e = \frac{\sum_{i=1}^{K} d_{e_i}}{K} \; ; \; \alpha_{e_{K+1}} = \frac{d_{e_{K+1}}}{\bar{d}_e} \; ; \; \bar{\alpha}_{K+1} = \frac{\sum_{e=1}^{n} \alpha_{e_{K+1}}}{n} \; ; \; d_{e_{K+1}} = \bar{\alpha}_{K+1} \bar{d}_e$$

Figure 12.8: Source Discarding Policies Formula

$$((t > t_{\min}) \wedge (r \geq r_{\min})) \vee (t > t_{\max})$$

Figure 12.9: Minimum Granted Results Policy Formula

12.7 Minimum Granted Results Policy

During our work on designing and implementing specialised meta-search engines we have developed a policy that we call Minimum Granted Results Policy (MGR), that satisfies two requirements:

1. The waiting policy must be able to abort the search process if a minimum results have been achieved and the response time has surpassed some desirable level.

2. Even if the minimum results have not been achieved a maximum time-out must be established.

These two requirements can be more formally expressed. The first one just says that if the delay t surpasses some minimum time tmin and the quality of the results already obtained r surpasses some minimum level rmin the search process must end. The second requirement says simply that the delay never must surpass some maximum level tmax. See the formula in Fig. 12.9.

Fig. 12.10[h!!] shows how this algorithm behaves in the same two environments described before. With normal network conditions (left graph) the system has the chance to harvest results even over the rmin level. When the delay arrives to tmin the search process is aborted. Under hostile network conditions (right graph) the tmin level of delay must be surpassed because the number of successful target engines is still not enough to fulfil a response. When rmin engines finish (or the value of pondering the engines that have finished reach rmin) the search process is terminated. The graphs does not show the case when tmax is reached but no rmin, this could happen under exceptionally bad network performance. We have tested this policy in an implementation of a news metasearch engine [144]. For this case we have used heuristic values for rmin, tmin and tmax, but we are studying how to automate this process by obtaining user's feed-back. This can be achieved implicitly, by analysing user actions after different levels of delay (abandoning the session, retyping the query, etc.), or explicitly, by letting users define themselves what they consider a good results level, an unacceptable delay, or how to ponder the different sources (this can be stored in user's profile).

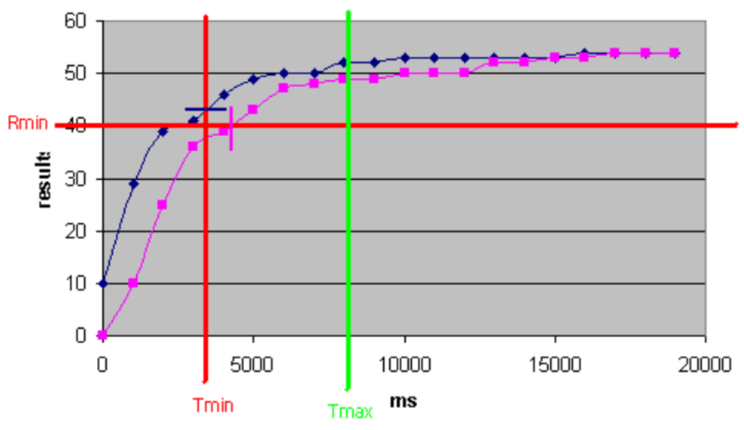


Figure 12.10: Minimum Granted Results Policy

12.8 Conclusions

Specialised distributed search is a natural link in the information feed-chain. However, some specialised search engines have response times often higher and more unpredictable than traditional Web-wide search systems like Google [23]. This constraints the performance and QoS of specialised distributed search systems and motivates the study of techniques to optimise the trade-off between delay and results quality. Aborting a search process to improve performance can seem a sin from a traditional information retrieval point of view, but if we consider that the recall value for the underlying engines is far from being 100%, and taking to consideration that most part of users usually consider only the first results, to abort the search proc-ess when results have reached some reasonable quality can be a good measure. We have applied the techniques described here to the development of a specialised meta-search engine that searches and retrieves newspaper news. Most part of the sources we have used in the practical application are on-line news services of the Spanish market, and have proved to be slow and unreliable. After some initial frustration we decided to apply the MGR policy described above. The experience of using the system improved notably, because the policy hides to the user the underlying engines performance and the network conditions.

Now we are working to automate some features of the system, that includes among other issues (automatic source discovery and configuration, automatic extraction of meta-data) the tuning of the parameters of the MGR policy. As we pointed out before, this automation can be done implicitly (observing user reactions under certain conditions) or explicitly (letting user to establish some of the parameters). We are also working on new techniques to improve the overall QoS, that includes different levels of caching and the study of some related issues like cache live times.

Part IV

General Conclusions

Chapter 13

Conclusions

Because the results of the work can be divided in different parts, corresponding to the different related research papers, I've chosen to keep their specific conclusions (see previous chapters). However, here I'm going to summarise these conclusions and to give some final comments.

13.1 Heterogeneous Data Models and Schemas: Semantic Integration

Within the general area of data integration, this thesis presents two contributions to the semantic integration research trend. In on hand, I present a novel approach to the problem of XML semantic integration. It aims to overcome some limitations of already existing solutions (e.g. [91][84][65][54]) applying an old idea of [97] related to the XML-Relational data integration. The idea consists on represent the general XML model, instead of some specific schemas, over constructs of another model (Relational Model in the case of [97], RDF and OWL in my case). This translation of XML to RDF allows loading XML instances and also multiple XSD schemas into an RDF-based repository, that can also host different ontologies.

Over this idea, I have implemented a schema-aware and ontology-aware XPath processor, that can be used to transparently resolve queries over XML instances bound to schemas that define inheritance hierarchies among types and element names, or related to ontologies that define relationships that are relevant for the queries evaluation. The materialization of the approach in the Contorsion API, and its usage in a plausible usage scenario like the Digital Rights Management domain, demonstrates its usefulness.

In the other hand, but also within the semantic integration area, I have contributed to the ontology alignment problem. The thesis presents a novel sutructure-based semantic similarity measure based on a matrix represention of nodes from an RDF labelled directed graph. The approach is based on the intuitive idea that similarity of two concepts can be defined in terms of how they relate to other concepts. We model these relationships with a vector space of n dimensions being n the number of selected standard predicates (e.g. *rdfs:subClassOf*, *rdfs:range* or *foaf:name*). We have adapted the algoritm in [20] to these idea to iteratively compute the similarities between two OWL ontologies. I have presented also an optimization of the algorithm to critically reduce its computational cost. The good results obtained in the tests performed over the Ontology Alignment Evaluation Initiative 2005 testsuite has proven the value of the approach.

13.2 Heterogeneous Query Interfaces: XML Query Tunneling

Also within the general area of data integration, this thesis contributes to the problem of dealing with heterogeneous query interfaces. I have suggested a strategy that allows redistributing an expressive user query (expressed in a XML-based data query language) over a set of autonomous and heterogeneous databases accessed through web forms. This is a new practical solution to an old problem of the data integration LAV (Local-As-View) approach. How a initial query, targeting the logical mediated schema, must be translated into queries over the different autonomous data sources.

The idea, that has recently been renamed by Thomas Kabisch [81] as "Query Tunneling", consists on using XML-related technologies for the reprocessing of the initial user query over the results returned by the different sources, that must be a superset of the results that satisfy the initial query. The usefulness of the approach, that theoretically does not improve the old solutions based in Datalog [51][119][56], relies in its applicability, that has been demonstrated in the development of a spanish news metasearch engine, and the Java Simple API for Web Information Integration (SAWII), that offers high level tools to the development of articulated wrappers for complex web form-chains and result pages.

13.3 A Final Comment

The integration of data from multiple heterogeneous sources is an old and well-known research problem for the database and AI research communities. However, the evolution of the World Wide Web and other distributed environments like Peer-to-Peer and the Grid has refuelled some research challenges of this area with a great practical importance.

While the amazing success of XML has clearly improved the interoperability of data and metadata in the digital environment, the recent success of not so recent semantic-rich modelling languages under the global name of *The Semantic Web Initiative* has raised a new opportunity and challenge to the data integration community. Ontologies, instead of schemas, are the new way to represent information domains. They are built with a rich set of constructs provided by the Semantic Web modelling languages like RDFS [126] and OWL [112].

Despite of XML and RDF derive from technologies more than 30 years old, and without discussing the reasons, it is clear that they are at the *centre of the stage*, changing the way to do a lot of things that have remained unchanged for a long time. Their impact has stimulated the imagination of a lot of people, and has generated a lot of initiatives. Most of these initiatives are not new, but have been revitalised in the new context. Some of these initiatives are those related to the dissemination and use of machine-readable metadata, that take profit from the benefits in terms of interoperability that XML and RDF offer. However, it is not clear where is the limit of the advantages of structured data, and not everybody agree in what is going to be the future scenario.

Bibliography

[1] Aberer, K., Cudré-Mauroux, P., and Hauswirth, M. The Chatty Web: Emergent Semantics Through Gossiping. In *Proceedings of the 12th International World Wide Web Conference* (2003). 53

[2] Adali, S., Candan, K. S., Papakonstantinou, Y., and Subrahmanian, V. S. Query caching and optimization in distributed mediator systems. In *SIGMOD Conference* (1996), pp. 137–148. 20, 113

[3] Alexaki, S., Christophides, V., Karvounarakis, G., and Plexousakis, D. The rdfsuite: Managing voluminous rdf description bases. Technical report, Institute of Computer Science, FORTH, Heraklion, Greece. See http://www.ics.forth.gr/proj/isst/RDF/RSSDB/rdfsuite.pdf. 13

[4] Alon, S. G. What can databases do for peer-to-peer? See http://citeseer.ifi.unizh.ch/653741.html. 53

[5] Amann, B., Beeri, C., Fundulaki, I., and Scholl, M. Ontology-based integration of xml web resources. In *Proceedings of the 1st International Semantic Web Conference (ISWC 2002),pages 117-131* (2002). 2, 52, 61

[6] Ashpole, B. Ontology translation protocol (ontrapro). In *E. Messina and A. Meystel, editors, Proceedings of Performance Metrics for Intelligent Systems (PerMIS'04)* (2004). 37

[7] Avnur, R., and Hellerstein, J. M. Eddies: continuously adaptive query processing. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data* (2000), pp. 261–272. 29

[8] Azuaje, F., and Bodenreider, O. Incorporating ontology-driven similarity knowledge into functional genomics: An exploratory study. In *BIBE* (2004), pp. 317–324. 3, 41

[9] Baeza-Yates, R., and Ribeiro-Neto, B. *Modern Information Retrieval.* Addison-Wesley-Longman Publishing co., 1999. 3, 7, 8, 11

[10] Baker, T. *A Grammar of Dublin Core.* D-Lib Magazine. October 2000. Volume 6 Number 10. ISSN 1082-9873, 2000. 3

[11] Berners-Lee, T. Information management: A proposal. *CERN DD/OC* (1989). 3

[12] Berners-Lee, T. Semantic web road map, 1998. See http://www.w3.org/DesignIssues/Semantic.html. 3, 12

[13] Berners-Lee, T. Why rdf model is different from the xml model. w3c draft september 1998, 1998. See http://www.w3.org/DesignIssues/RDF-XML.html. 3

[14] Berners-Lee, T. Transcript of tim berners-lee's talk to the lcs 35th anniversary celebrations, cambridge massachusetts, 1999/april/14, 1999. 3, 10

[15] Berners-Lee, T., Cailliau, R., Groff, J.-F., and Pollermann, B. World-wide web: The information universe. *Electronic Networking: Research, Applications and Policy, Vol 1 No 2, Meckler, Westport CT* (1992). 3, 12

[16] Berners-Lee, T., Fielding, R., and Masinter, L. Internet rfc 2396, uniform resource identifiers (uri): Generic syntax. 14

[17] Berners-Lee, T., Hendler, J., and Lassila, O. The semantic web. *Scientific American* (May 2001). 3, 12

[18] Bernstein, P., Giunchiglia, F., Kementsietsidis, A., Mylopoulos, J., Serafini, L., and Zaihrayeu, I. Data management for peer-to-peer computing: A vision. In *Workshop on the Web and Databases, WebDB* (2002). 53

[19] Bisson, M. Learning in fol with a similarity measure. In *Proceedings of the 10th American Association for Artificial Intelligence conference, San Jose (CA US), pages 82-87* (1992). 3, 39, 86

[20] Blondel, V. D., Gajardo, A., Heymans, M., Senellart, P., and Dooren, P. V. A measure of similarity between graph vertices: Applications to synonym extraction and web searching. *SIAM Rev. 46*, 4 (2004), 647–666. 3, 5, 40, 41, 43, 44, 45, 85, 87, 89, 90, 97, 98, 127

[21] Blue, A. Davis drives home the importance of being knowledge based. *Information Outlook, 2(5): 39.* (1997). 7

[22] Brickley, D., and Guha, R. Resource description framework (rdf)schema specification (proposed recommendation), w3c (world wide web consortium), 1999. See http://www.w3.org/TR/1999/PR-rdf-schema-19990303. 3, 12

[23] Brin, S., and Page, L. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems* (1998). 3, 123

[24] Callan, J. Distributed information retrieval. *Advances in information retrieval, chapter 5, pages 127-150. Kluwer Academic Publishers* (2000). 9

[25] Chang, W. A discussion of the relationship between rdf-schema and uml, 1998. W3C Note 04-Aug-1998. 3

[26] Chawathe, S. S., Garcia-Molina, H., Hammer, J., Ireland, K., Papakonstantinou, Y., Ullman, J. D., and Widom, J. The tsimmis project: Integration of heterogeneous information sources. In *IPSJ* (1994), pp. 7–18. 20, 113

[27] Costa, Paulo C. G.; Laskey, K. B., and Laskey, K. J. Pr-owl: A bayesian framework for the semantic web. In *Workshop on Uncertainty Reasoning for the Semantic Web (URSW 2005), held at the Fourth International Semantic Web Conference (ISWC 2005). November, 7th 2005, Galway, Ireland.* (2005). 51, 52

[28] Dublin core. See http://dublincore.org/. 3, 7

[29] Iso/iec, iso/iec 2nd edition fcd 21000-2 - digital item declaration. 79

[30] Doan, A., Domingos, P., and Halevy, A. Y. Reconciling schemas of disparate data sources: A machine-learning approach. In *SIGMOD Conference* (2001). 41

[31] Doan, A., Domingos, P., and Halevy, A. Y. Learning to match the schemas of data sources: A multistrategy approach, 2003. 50

[32] Doan, A., Madhavan, J., Domingos, P., and Halevy, A. Learning to map between ontologies on the semantic web. In *The Eleventh International WWW Conference, Hawaii, US, 2002.* (2002). 37, 41, 86

[33] Dreilinger, D. Integrating heterogeneous www search engines, 1995. Masters Thesis, Colorado State University. 3, 10, 113, 115

[34] Duschka, O. M., and Genesereth, M. R. Answering recursive queries using views. In *PODS '97: Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems* (New York, NY, USA, 1997), ACM Press, pp. 109–116. 20, 113

[35] Ehrig, M., and Euzenat, J. Relaxed precision and recall for ontology matching. See http://km.aifb.uni-karlsruhe.de/ws/intont2005/intontproceedings.pdf. 37, 86

[36] Ehrig, M., and Staab, S. Qom - quick ontology mapping. In *Proc. of the Third International Semantic Web Conference (ISWC2004)* (2004). 37, 86

[37] Euzenat, J., and Valtchev, P. Similarity-based ontology alignment in owl-lite. In *Proc. of ECAI 2004, pages 333–337, Valencia, Spain, August 2004* (2004). 37, 39, 86

[38] Fleiss, J. L. Statistical methods for rates and proportions. isbn: 0-471-52629-0, 1973. 41

[39] Rdfweb: Friend of a friend (foaf). See http://rdfweb.org/foaf/. 3

[40] Forgy, L. Rete: A fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence* (1982). 77

[41] Friedman, M., Halevy, A. Y., and Millstein, T. Navigational plans for data integration. In *AAAI '99/IAAI '99: Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence* (Menlo Park, CA, USA, 1999), American Association for Artificial Intelligence, pp. 67–73. 20

[42] G. Karvounarakis, V. Christophides, D. P., and Alexaki, S. Querying community web portals, 2000. Technical report, Institute of Computer Science, FORTH, Heraklion, Greece. See http://www.ics.forth.gr/proj/isst/RDF/RQL/rql.pdf. 13

[43] Gallaire, H., and Minker, J., Eds. *Logic and Data Bases.* Perseus Publishing, 1978. 10

[44] Gangemi, A., Guarino, N., Masolo, C., Oltramari, A., and Schneider, L. Sweetening ontologies with dolce. In *13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02), volume 2473 of Lecture Notes in Computer Science, page 166 ff, Sig uenza, Spain, Oct. 1-4* (2002). 36, 48, 49

[45] Garcia, R., and Delgado, J. Brokerage of intellectual property rights in the semantic web. In *Proc. Semantic Web Working Symposium. Stanford University, California, pp. 245-260* (2001). 76

[46] Garcia, R., Gil, R., and Delgado, J. Intellectual property rights management using a semantic web information system. In *OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2004. Springer-Verlag, LNCS Vol. 3291, pp. 689-704* (2004). 76

[47] Gil, R., Garcia, R., and Delgado, J. Delivery context negotiated by mobile agents using cc/pp. *Int. Conference on Mobile Agents for Telecommunication Applications, MATA 03. LNCS, Vol. 2881, pp 99-110. Springer-Verlarg* (2003). 76

[48] Gil, R., Garcia, R., and Delgado, J. An interoperable framework for ipr using web ontologies. In *Legal Ontologies and Artificial Intelligence Techniques Workshop, LOAIT 2005. To be published in the International Association for Artificial Intelligence and Law Workshop Series* (2005). 76

[49] Guha, R., and McCool., R. Tap: A semantic web platform, 2003. See http://tap.stanford.edu/tap.pdf. 3, 13

[50] Haas, L. M., Kossmann, D., Wimmers, E. L., and Yang, J. Optimizing queries across diverse data sources. In *Proceedings of the Twenty-third International Conference on Very Large Databases* (Athens, Greece, 1997), VLDB Endowment, Saratoga, Calif., pp. 276–285. See http://citeseer.ist.psu.edu/article/haas97optimizing.html. 20, 113

[51] Halevy, A. Y. Answering queries using views: A survey. *VLDB Journal: Very Large Data Bases 10*, 4 (2001), 270–294. 24, 101, 128

[52] Halevy, A. Y., Ives, Z., Mork, P., and Tatarinov, I. Piazza: Data management infrastructure for semantic web applications. In *Proc. of the 12th Intl. World Wide Web Conf.* (2003). 30, 53

[53] Halevy, A. Y., Ives, Z., Suciu, D., and Tatarinov, I. Schema mediation in peer data management systems. In *Proceedings of ICDE* (2003). 53

[54] Halevy, A. Y., Ives, Z. G., Mork, P., and Tatarinov, I. Piazza: Data management infrastructure for semantic web applications. In *Proceedings of the 12th International World Wide Web Conference* (2003). 2, 52, 61, 127

[55] Halevy, A. Y., Mendelzon, A. O., Sagiv, Y., and Srivastava, D. Answering queries using views. In *Proceedings of the 14th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems* (San Jose, Calif., 1995), pp. 95–104. 25, 105

[56] Halevy, A. Y., Rajaraman, A., and Ordille, J. J. Querying heterogeneous information sources using source descriptions. In *Proceedings of the Twenty-second International Conference on Very Large Databases* (Bombay, India, 1996), VLDB Endowment, Saratoga, Calif., pp. 251–262. See http://citeseer.ist.psu.edu/levy96querying.html. 20, 24, 29, 52, 113, 128

[57] Halevy, A. Y., Rajaraman, A., and Ullman, J. D. Answering queries using limited external query processors (extended abstract). In *Proceedings of the fifteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems* (1996), ACM Press, pp. 227–237. 29

[58] Hopcroft, J. E., and Karp, R. An $\mathcal{O}(n^{5/2})$ algorithm for maximum matching in bipartite graphs. *SIAM J. Comput. 4* (1973), 225–231. 39, 86

[59] Horrocks, I., and Patel-Schneider, P. F. Reducing owl entailment to description logic satisfiability. In *Proc. of the 2003 Description Logic Workshop (DL 2003), volume 81 of CEUR, pages 1-8* (2003). 3, 15

[60] Horrocks, I., Patel-Schneider, P. F., and van Harmelen, F. From shiq and rdf to owl: The making of a web ontology language. *J. of Web Semantics* (2003). 3, 15

[61] Horrocks, I., and Sattler, U. Ontology reasoning in the shoq (d) description logic. In *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence, pages 199-204* (2001). 3, 15

[62] Horrocks, I., Sattler, U., and Tobies, S. Practical reasoning for very expressive description logics. *Proc. of the 6th Int. Conf. on Logic for Programming and Automated Reasoning (LPAR 99), number 1705 in Lecture Notes in Artificial Intelligence, pages 161-180. Springer* (1999). 3, 15, 65, 73

[63] Horrocks, I., Sattler, U., and Tobies, S. Practical reasoning for very expressive description logics. *J. of the Interest Group in Pure and Applied Logic* (2000). 3, 15

[64] Hu, W., Jian, N., Qu, Y., and Wang, Y. Gmo: A graph matching for ontologies. In *Integrating Ontologies* (2005). See http://CEUR-WS.org/Vol-156/paper7.pdf. xiv, 3, 40, 41, 43, 45, 87, 91, 92, 96, 97

[65] I. Cruz, H. X., and Hsu, F. An ontology-based framework for xml semantic integration. In *Proceedings of the Eighth International Database Engineering and Applications Symposium. IDEAS'04. July 7-9, 2004 Coimbra, Portugal* (2004). 2, 52, 61, 64, 71, 127

[66] R. iannella. open digital rights language (odrl), version 1.1. world wide web consortium 2002 (w3c note). See http://www.w3.org/TR/odrl. 78

[67] Ichise, R., Takeda, H., and Honiden, S. Rule induction for concept hierarchy alignment. In *Proceedings of the Workshop on Ontology Learning at the 17th International Joint Conference on Artificial Intelligence (IJCAI) , 2001.* (2001). 41

[68] Xml information set (second edition) w3c recommendation 4 february 2004. See http://www.w3.org/TR/xml-infoset/. 3, 30, 61, 64, 66

[69] Xml path language (xpath) 2.0 w3c working draft 23 july 2004. See http://www.w3.org/TR/xpath20/. 3, 31, 66, 67, 68

[70] Iso/iec, iso/iec fcd 21000-4 - intellectual property management and protection components. 79

[71] Ives, Z. G., Florescu, D., Friedman, M., Halevy, A. Y., and Weld, D. S. An adaptive query execution system for data integration. In *ACM SIGMOD Record, Proceedings of the 1999 ACM SIGMOD international conference on Management of data SIGMOD '99, Volume 28 Issue 2* (1999), pp. 299–310. 29

[72] Ives, Z. G., Halevy, A. Y., Weld, D. S., Florescu, D., and Friedman, M. Adaptive query processing for internet applications. *IEEE Data Engineering Bulletin 23*, 2 (2000), 19–26. 29

[73] J. Broekstra, A. K., and Harmelen, F. Sesame: An architecture for storing and querying rdf data and schema information, 2001. 13

[74] J. Delgado, I. G., and Garcia, R. Use of semantic tools for a digital rights dictionary. *E-Commerce and Web Technologies: 5th International Conference, 2004. K. Bauknecht, K. and M. Bichler and B. Proll. LNCS Volume 3182 (338-347) Springer-Verlag* (2004). 82

[75] Jaxen: Universal java xpath engine. See http://jaxen.org/. 3, 76

[76] Jena 2. a semantic web framework. See http://www.hpl.hp.com/semweb/jena.htm. 3, 70, 76

[77] D. reynolds. jena 2 inference support. See http://jena.sourceforge.net/inference/. 3, 77

[78] Routefinding over rdf geodata with jena2 rdql. See http://chimpen.com/things/archives/001182. 78

[79] Jiang, J., and Conrath, D. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of the International Conference on Research in Computational Linguistics* (1997). 3, 41

[80] Project jxta homepage. See http://www.jxta.org/. 13

[81] Kabisch, T., and Neiling, M. Wrapping of web sources with restricted query interfaces by query tunneling. In *Proc. of InterDB 2005. International Workshop on Database Interoperability* (2005). v, 5, 52, 113, 128

[82] Kementsietsidis, A., Arenas, M., and Miller, R. Mapping data in peer-topeer systems: Semantics and algorithmic issues. In *Proceedings of VLDB* (2003). 53

[83] Khare, R., and Rifkin, A. Weaving a web of trust. *World Wide Web Journal, Volume 2, Number 3, Pages 77-112, Summer 1997* (1997). 14

[84] Klein, M. C. A. Interpreting xml documents via an rdf schema ontology. In *Proceedings of the 13th International Workshop on Database and Expert Systems Applications (DEXA 2002),pages 889-894* (2002). 52, 61, 127

[85] Kobayashi, M., and Takeda, K. Information retrieval on the web. *ACM Computing Survey* (2000). xiii, 3, 116, 117

[86] Lagoze, C., and Sompel, H. The open archives initiative: Building a low-barrier interoperability framework. *ACM/IEEE Joint Conference on Digital Libraries* (2001). 3

[87] Lakshmanan, L. V., and Sadri, F. Interoperability on xml data. In *Proceedings of the 2nd International Semantic Web Conference (ICSW 03)* (2003). 2, 52, 61

[88] Lawrence, S., and Giles, L. Searching the world wide web. *Science* (1998). 3, 10, 115

[89] Leacock, C., and Chodorow, M. Combining local context and wordnet similarity for word sense identification. *WordNet: An Electronic Lexical Database 49(2)* (1998), 265–283. 40, 97

[90] Lee, J. H., Kim, M. H., and Lee, Y. J. Information retrieval based on conceptual distance in is-a hierarchies. *Journal of Documentation 49(2)* (1993), 188–207. 40, 97

[91] Lehti, and Fankhauser. Xml data integration with owl: Experiences & challenges. *SAINT 160-170* (2004). 2, 60, 127

[92] Lenat, D. B. CYC: A large-scale investment in knowledge infrastructure. *Communications of the ACM 38*, 11 (1995), 33–38. 36, 48, 49

[93] LENZERINI, M. Data integration: a theoretical perspective. In *PODS '02: Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems* (New York, NY, USA, 2002), ACM Press, pp. 233–246. 20

[94] LIN, D. An information-theoretic definition of similarity. In *Proc. 15th International Conf. on Machine Learning* (1998), Morgan Kaufmann, San Francisco, CA, pp. 296–304. 3, 41

[95] LIN, D. Review of wordnet an electronic lexical database, 1998. See citeseer.ist.psu.edu/lin98review.html. 48

[96] LOSEE, R. *Text Retrieval and Filtering: Analytic Models of Performance.* Kluwer, Boston, 1998. 116

[97] M. YOSHIKAWA, T. AMAGASA, T. S., AND UEMURA, S. Xrel: A path-based approach to storage and retrieval of xml documents using relational databases. *ACM Transactions on Internet Technology, Vol. 1, No. 1, June 2001* (2001). 2, 52, 61, 127

[98] MANOLESCU, I., FLORESCU, D., AND KOSSMANN, D. K. Answering XML queries over heterogeneous data sources. In *Proceedings of the 27th International Conference on Very Large Data Bases* (2001), pp. 241–250. 20, 113

[99] Mapforce. visual data integration and web services implementation tool, 2006. See http://www.altova.com/products/mapforce/data_mapping.html. 32

[100] MARCHIONINI, G. *Information Seeking in Electronic Environments.* Cambridge University Press, 1995. 7

[101] Mpeg-7. See http://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm. 2, 8, 113

[102] NEJDL, W., AND ET AL., B. W. Edutella: A p2p networking infrastructure based on rdf, 2001. 13

[103] NEJDL, W., WOLF, B., QU, C., DECKER, S., NAEVE, M. S. A., NILSSON, M., PALMER, M., AND RISCH, T. Edutella: A p2p networking infrastructure based on rdf, 2001. 53

[104] NG, W. S., OOI, B. C., TAN, K.-L., AND ZHOU, A. Peerdb: A p2p-based system for distributed data sharing. In *Proceedings of the Intl. Conf. on Data Engineering (ICDE)* (2003). 53

[105] NILES, I., AND PEASE, A. Towards a standard upper ontology. In *FOIS '01: Proceedings of the international conference on Formal Ontology in Information Systems* (New York, NY, USA, 2001), ACM Press, pp. 2–9. 36, 48

[106] NOY, N. F. Semantic integration: a survey of ontology-based approaches. *SIGMOD Rec. 33*, 4 (December 2004), 65–70. 3, 13, 37, 48

[107] Noy, N. F., and Musen, M. A. PROMPT: Algorithm and tool for automated ontology merging and alignment. In *AAAI/IAAI* (2000), pp. 450–455. 37

[108] Ontology alignment evaluation initiative. 2005 campaign, 2005. See http://oaei.inrialpes.fr/2005/. 37, 39, 86, 96

[109] The open archives initiative. See http://www.openarchives.org/. 3, 104

[110] The open archives initiative protocol for metadata harvesting. See http://www.openarchives.org/OAI/openarchivesprotocol.html. 3

[111] Foldoc, free on-line dictionary of computing. http://wombat.doc.ic.ac.uk/foldoc/. 7

[112] Owl web ontology language overview. w3c recommendation 10 february 2004. See http://www.w3.org/TR/owl-features/. 2, 3, 14, 36, 37, 40, 52, 86, 97, 128

[113] Page, L., Brin, S., Motwani, R., and Winograd, T. The pagerank citation ranking: Bringing order to the web. *Stanford Digital Library Technologies Project* (1998). 3

[114] Papadimitriou, C. H., and Steiglitz, K. *Combinatorial optimization: algorithms and complexity.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1982. 39, 86

[115] Papakonstantinou, Y., Gupta, A., Garcia-Molina, H., and Ullman, J. D. A query translation scheme for rapid implementation of wrappers. In *4th Intl. Conf. on Deductive and Object-Oriented Databases; LNCS 1013* (Singapore, 1995), vol. Extended version available at: ftp://ftp.db.stanford.edu/pub/papakonstantinou/1995/querytran-extended.ps, Springer Berlin, Heidelberg, New York, pp. 319–344. 29

[116] Papakonstantinou, Y., and Vassalos, V. Architecture and implementation of an xquery-based information integration platform. See http://citeseer.ist.psu.edu/612914.html. 34

[117] Patel-Schneider, P. F., and Simeon, J. The yin/yang web:xml syntax and rdf semantics. In *Proceedings of the 11th International World Wide Web Conference (WWW2002),pages 443-453* (2002). 2, 52, 61

[118] Peig, E., Delgado, J., and Perez, I. Metadata interoperability and meta-search on the web. In *Proceedings of the International Conference on Dublin Core and Metadata Applications (DC-2001)* (2001). 103, 116

[119] Pottinger, R., and Halevy, A. Y. MiniCon: A scalable algorithm for answering queries using views. *VLDB Journal: Very Large Data Bases 10*, 2–3 (2001), 182–198. 24, 34, 52, 128

[120] R. Guha, R. M., and Miller, E. Semantic search, 2003. See http://tap.stanford.edu/ess.pdf. 3

[121] Rada, R., Mili, H., and Bicknell, E. Development and application of a metric on semantic nets. *IEEE Transactions on Systems, Man and Cybernetics* (1989), 17–30. 40

[122] Rahm, E., and Bernstein, P. A. A survey of approaches to automatic schema matching. *VLDB Journal: Very Large Data Bases 10*, 4 (2001), 334–350. 3, 39, 50, 86

[123] Iso/iec, iso/iec is 21000-6 - rights data dictionary. 79

[124] Resource description framework. See http://www.w3.org/RDF/. 2, 13, 52

[125] rdfdb query language. See http://www.guha.com/rdfdb/query.html. 3, 70

[126] Rdf vocabulary description language 1.0: Rdf schema. See http://www.w3.org/TR/2003/WD-rdf-schema-20030123/. 14, 36, 52, 128

[127] Rdql - a query language for rdf. w3c member submission 9 january 2004. See http://www.w3.org/Submission/RDQL/. 3, 70, 71

[128] Iso/iec, iso/iec is 21000-5 - rights expression language. 79

[129] Resnik, P. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *Journal of Artificial Intelligence Research 11* (1999), 95–130. 3, 41

[130] Reynolds, D. Rdf-qbe: a semantic web building block. *HP-Lab* (2003). 3, 13

[131] Rijsbergen, C. *Information Retrieval.* Butterworths, 1979. 8

[132] Rpath - rdf query language proposal. See http://web.sfc.keio.ac.jp/ km/rpath-eng/rpath.html. 52, 61, 71

[133] Rust, G., and Barlas, C. The mpeg-21 rights data dictionary. *IEEE Transactions on Multimedia, volume 7 number 2* (2005). 78

[134] Salton, G. The smart retrieval system. *Experiments in Automatic Document Processing* (1971). 87

[135] Selberg, E., and Etzioni, O. The metacrawler architecture for resource aggregation on the web. *IEEE Expert* (1997). 3, 10, 113, 115, 116

[136] Soap. See http://www.w3.org/TR/SOAP/. 107

[137] Sompel, H., and Lagoze, C. The santa fe convention of the open archives initiative. *D-Lib Magazine, 6(2), February 15, 2000* (2000). 3

[138] Sparql query language for rdf. w3c working draft 12 october 2004. See http://www.w3.org/TR/rdf-sparql-query/. 3, 70

[139] Inkling: Rdf query using squishql. See http://swordfish.rdfweb.org/rdfquery/. 3, 70

[140] Search/retrieve web service (srw). See http://lcweb.loc.gov/z3950/agency/zing/srw/specifications.html. 3

[141] Stoilos, G., Stamou, G., Tzouvaras, V., Pan, J. Z., and Horrocks, I. Fuzzy owl: Uncertainty and the semantic web. In *OWL: Experiences and Directions Workshop* (2005). 51, 52

[142] Semantic web. See http://www.w3.org/2001/sw. 12

[143] Html tidy. See http://www.w3.org/People/Raggett/tidy/. 32, 111

[144] Tous, R., and Delgado, J. Advanced metasearch of news in the web. In *Proceedings of the International Conference on Electronic Publishing 2002. (ElPub 2002)* (2002). 106, 116, 122

[145] V. Raghavan, P. B., and Jung, G. A critical investigation of recall and precision as measures of retrieval system performance. In *ACM Transactions on Information Systems, 7(3):205-229* (2001). 116

[146] W. Meng, C. Y., and Liu, K. Building efficient and effective metasearch engines. *ACM Computing Surveys* (2002). 115

[147] Rdf/xml syntax specification (revised). w3c recommendation 10 february 2004. See http://www.w3.org/TR/rdf-syntax-grammar/. 3, 61, 64

[148] Wu, Z., and Palmer, M. Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics* (Morristown, NJ, USA, 1994), Association for Computational Linguistics, pp. 133–138. 40, 97

[149] X. Wang, T. D., Wragg, B., and Paramasivam, M. The mpeg-21 rights expression language. *IEEE Transactions on Multimedia volume 7 number 2* (2005). 78

[150] Xcql. See http://www.loc.gov/z3950/agency/zing/cql/xcql.html. 4

[151] Extensible markup language (xml). See http://www.w3.org/XML/. 2, 11

[152] Xquery 1.0 and xpath 2.0 data model. w3c working draft 23 july 2004. See http://www.w3.org/TR/xpath-datamodel/. 3, 31, 33, 66

[153] Xquery 1.0 and xpath 2.0 formal semantics. w3c working draft 3 june 2005. See http://www.w3.org/TR/xquery-semantics/. 3, 66

[154] Xquery 1.0: An xml query language. w3c working draft 04 april 2005. See http://www.w3.org/TR/xquery/. 2, 3, 4, 33, 52, 66, 104, 107

[155] Xsl transformations (xslt) version 2.0. w3c working draft 4 april 2005. See http://www.w3.org/TR/xslt20/. 3, 66

[156] Z39.50. See ftp://ftp.loc.gov/pub/z3950/official/part1.pdf. 3